order number:. . .

Thesis submitted to the

UNIVERSITY OF MOHAMED BOUDIAF - MSILA

FACULTY OF MATHMATICS AND COMPUTER SCIENCE DEPARTEMENT OF COMPUTER SCIENCE

In partial fulfillment of the requirements for the degree of

Master in **Computer science**

By

Benslimane Echyma
Gherabi Amira

Title of the thesis

Vision based vehicle counting at crowded intersections

Under the supervision of

Ghemougui Abdessattar

Composition of the jury

Kamel Mohamed	University of Msila	President
Ghemougui Abdessettar	University of Msila	Reporter
Boughrara Seddik	University of Msila	Examiner

June 2022

Acknowledgements

This study would have been impossible without the support and encouragement of many people.
This was a journey worthwhile taking.

First of all, we would like to thank **Allah**, the merciful, for giving us the health and the strength to finish this thesis.

Second, we would like to thank our supervisor **Mr.Abdessattar Ghemougui** , for his insightful direction and support during the period of writing this thesis.

Finally, our deep and gratitude to our **Family** and **Friends** for their help and support they give us.
Thank you, All.

Abstract

Traffic congestion has a significant negative impact on the environment and on people, and one of the most important reasons is poor control of traffic signals. Vehicle counting in crowded intersections is a crucial element of any traffic optimization solution. In this work, we propose a computer vision based vehicle counting method. We adopts YOLO-v3 as vehicle detector, we added a series of post-processing mechanisms to achieve robust vehicle counting. We created framework to asemi-utomatically generate a dedecated dataset for vehicle counting, this dataset can be later used as training data for new maching learning based solutions.

Keywords: Computer vision; Intelligent traffic signal control system; Vehicle counting.

الازدحام المروري له تأثير سلبي كبير على البيئة وعلى الناس ومن اهم اسباب هو سوء تحكم في اشارة المرور. يعد حساب المركبات في التقاطعات عنصرًا حاسمًا في أي حل لتحسين حركة المرور. في هذا العمل ، نقترح طريقة عد المركبات القائمة على رؤية الكمبيوتر. يعتمد إطارنا YOLO ككاشف للمركبة ، أضفنا سلسلة من آليات ما بعد المعالجة لتحقيق عد قوي للمركبة. استخدمنا إطار العمل الخاص بنا لإنشاء مجموعة بيانات مخصصة لحساب المركبات تلقائيًا ، ويمكن استخدام مجموعة البيانات هذه لاحقًا كبيانات تدريب لحل جديد قائم على التعلم الآلي.

الكلمات المفتاحية: رؤية الحاسوب ؛ نظام ذكي للتحكم في إشارات المرور؛ عد المركبات .

Contents

List of abbreviations

TSC *Traffic Signal Control*

TRANSYT *Traffic Network Study Tool*

MOVA *Microprocessor Optimised Vehicle Actuation*

UTOPIA *Urban Traffic Optimization by Integrated Automation*

OPAC *Optimised for Adaptive Control Policies*

ITS *Intelligent traffic system*

CV *Computer Vision*

AI *Artificial Intelligence*

MIT *Massachusetts Institute of Technology*

OCR *Optical character recognition*

ANPR *automatic number plate recognition*

Pixel *PIcture ELement*

RGB *"Red-Green Blue"*

CMYK *Cyan magenta yellow black*

HSV *Hue saturation value*

Lab *Lightness Red/Green Blue/Yellow*

SIFT *Scale-Invariant Feature Transform*

HOG *Histogram of oriented gradients*

SVM *Support vector machine*

DNN *Deep Neural Network*

YOLO *You only look once*

RCNN *Region Based Convolutional Neural Networks*

MS COCO *Microsoft Common Objects in Context*

CNN *Convolutional Neural Networ*

IoU *Intersection of union*

ILSVRC *ImageNet Large Scale Visual Recognition Challenge*

BB *Bounding Box*

NMS *Non maximum supperssion*

List of Figures

List of Tables

General introduction

In the traffic system, road users are continuously moving, causing road congestion, causing huge social and economic losses, the estimation of the number of traffic vehicles also reflects the number of vehicles on the road, such as lane occupancy, congestion, these congestions are due to traffic caused by poor management. The most common strategies for urban traffic control can be divided into three types: non-induced based on fixed timing systems, semi-induced system based on sub-sensors, and induced control systems based on sensors on secondary roads and main roads. However, they have many drawbacks, such as long wait times (in green lights on empty roads and red lights on busy roads) or faulty sensors

In order to facilitate transportation and reach to destination as quickly as possible, an adequate level of control system should be ensured according to the flow of traffic. Therefore, traffic system control efficiency is widely considered as valid performance indicator to measure a country's transportation system and economy development level. In the developing world, societies are becoming more aware of the importance of improving traffic roads where the most common concern in managing urban traffic areas and road networks is Traffic Signal Control Systems (TSCS) which contribute to reduced congestion, reduced fuel consumption and reduced pollution. Several TCSCs have been developed and based on dynamic programming optimization procedures, which allow for continuous incremental adjustments in real time called intelligent TSCS and which exploit the availability of big data, computing power, and advanced methods to drive the development of intelligent transportation including artificial intelligence methods, computer vision in particular.

In order to achieve TSCS optimization we need traffic flow information. The number of vehicles in roads and intersection is a key pice of information. Therefore, vehicle counting in intersections is a key element of any traffic optimization solution. In this work we propose a computer vision and deep learning based solution to solve the problem of vehicle counting. the number of vehicles is then sent to the traffic control system where it will be used as input to traffic optimization algorithms.

Motivation

People are forced to spend hours in traffic congestions, wasting valuable time, which consumes energy and fuel. It has a significant negative impact on the environment by thus pollutant emissions from vehicles. Adding to the congestion is inefficient traffic signal control; each intersection is handled individually. So what are the main solutions to reduce congestion, save energy and improve traffic signal control systems?

Objectif

Our main goal of this work is to propose a vision based solution to the problem of vehicle counting in roads and intersections.

Thesis structure

This thesis is divided into four chapters organized as follows: The first chapter introduces traffic control system and traditional solutions, the second chapter, is a general introduction to computer vision, basic image processing and object detection. In chapter three presents the state-of-the-art solutions to the vehicle counting problem. In chapter Four, we present our approach for vehicle detection and counting.

Chapter 1

Traffic Optimization

1.1 Introduction

Transportation is an integral part of any community as it connects different regions and helps people to move easily between different destinations. Advances in transportation have made possible changes in the way societies are organized. The rapid increase in population has accelerated the growth of the number of vehicles and hence the complexity of transportation characteristics such as different types of drivers, pedestrians, cyclists, vehicles and road infrastructure. As a result, the demand for traffic is growing rapidly and continues to exceed transportation capacity. To better meet the traffic demand, it is necessary to upgrade the existing traffic systems.

Traffic refers to the movement of objects which are vehicles, people, etc., along roads in a particular area for a particular purpose.

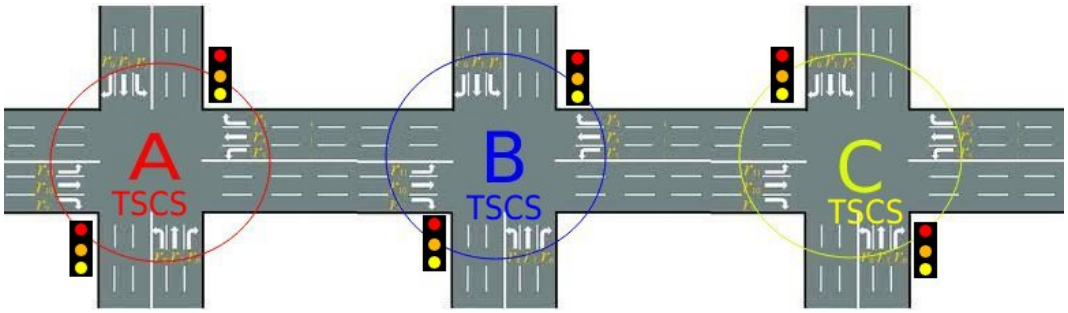


Figure 1.1: Representation of traffic signal control systems

The traffic system represent the minimal unit of traffic where in

figure 1.1 each one of A or B or C represent a traffic system, it contains all objects that can physically interact with each other, all objects in the traffic system must be controlled together. It can be divided in two types :

Figure 1.2: Different types of objects in a traffic system

- **moving objects:** cars, trucks, trams, trains, buses, bicycles, pedestrians, birds, even un-manned objects like ball passing the road;
- **steady objects:** parked cars, roadblocks, holes on the road ahead, trees in the side of road, lamppost.

The transportation system is currently facing several challenges and there is a need to decrease travel time and delays, improving passenger safety and reducing traffic exhaust emissions. In this chapter we will introduce some traffic signals control systems and stand on their point of weakness.

1.2 Traffic Signal Control Systems (TSCS)

A signaling electronic device located at intersections, crossroads or pedestrian crossing that control the sequence of the signals to regulate traffic movements.

A TSC system commonly consists of three signals, they control the traffic by signaling when vehicls have to stop and when they can go. Where

the red signal indicates that incoming vehicles have to stop, the green signal indicates that the vehicles are allowed to travel through the intersection if it is safe. The green arrow pointing right or left means that vehicles are allowed to make a protected turn. The amber warning signal comes after the green signal, it indicates that the traffic signal is about to turn red, the vehicles have to stop if possible. When the red and amber signals are shown at the same time, the vehicles have to completely stop.

For pedestrians, there are only two signals: a red signal, which means pedestrians have to stop, and a green signal, indicating that pedestrians can cross the road.[49]

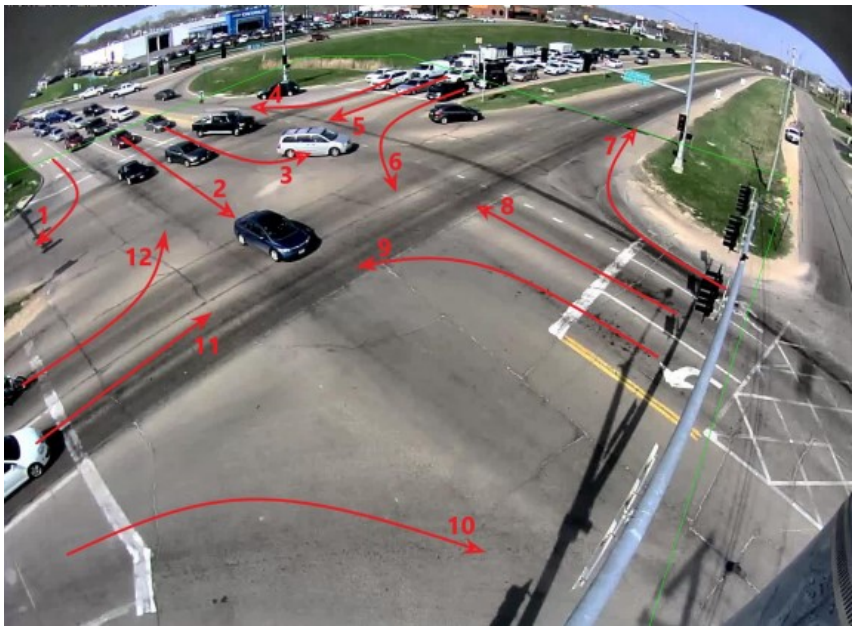


Figure 1.3: Possibles movements at an intersection [31]

This Figure 1.3 represents the movements of interest at one intersection. Each arrow line indicates one movement The TSC deployed at an intersection implements traffic signal timing to control vehicles, bicyclists, pedestrians, and other traffic participants safely passing through the intersection. Traffic signal timing includes deciding the sequence of movements and allocating green time to each group of movements at a signalized intersection. Pedestrians, cyclist and other users also should be taken into account when designing signal timings.

Intersections are one of the worst places to cause traffic congestion.

Therefore, researchers are trying to update traffic control equipment or adjust the schedule of signal signals to improve traffic efficiency. As a result, the following benefits can be achieved:

- Raising the efficiency of the people movement in daily life;
- reducing waiting time at intersections;
- Improving safety on the go by reducing the risk of accidents;
- Fuel saving and environmental protection;
- Reducing traffic congestion problems.

1.3 Conventional traffic signal control systems

The first traffic signal control system was used in London in 1868. This system applies manually operated traffic signals to prevent accidents by assigning a right-of-way to alternate vehicles. [13] Currently, there exists three types of TSCS methods [27]: non-induced, the semi-induced and fully induced systems

1.3.1 Non induced control system

In this family of TSC parameters are preset and fixed during the period, based on historical data. It is very suitable for the intersections where traffic demand is consistent in every day, such as intersections in areas of the city center. Therefore, by applying this type of system to the control, the following advantages can be achieved:

- Efficient coordination between adjacent intersections. Because all parameters can be set in advance, for example, the beginning and ending of the green time at adjacent intersections;
- immunize against the problems caused by the failure of the sensor;
- Cost effectiveness, it does not take much cost and training to install and maintain the system.

1.3.2 Semi induced control system

.

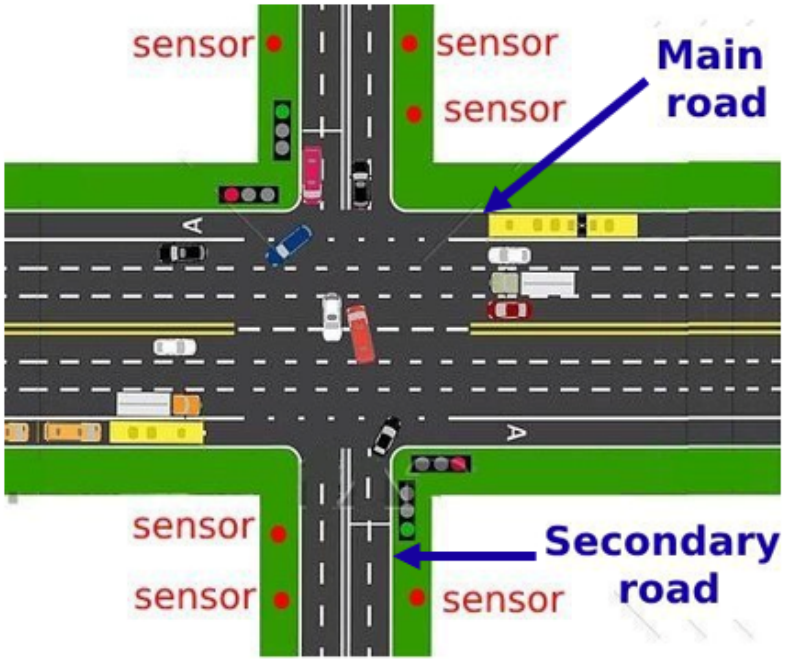


Figure 1.4: A semi-induced system representation

In this type of system as showing in Figure 1.4 the control center focuses on the highest traffic roads (highway or not prompted road). For small roads, the right of way is given based on informations transmitted by sensors placed on those roads. This control system is very suitable for intersections in the arterial street, which are either main or secondary. Both classes carry longer-distance flows between important centers of activity. Arteries are designed as the backbone of the traffic network and must be designed to provide the highest level of service. Where there is a large volume of traffic in the main road and a small demand in the lane. The following advantages can be obtained by applying this type of control system:

- It is very easy for this system to be applied effectively in a coordinated traffic control system;
- It can reduce the delay for traffic in highways;

- The failure of sensors does not affect the main road, as there are no sensors on these roads.

1.3.3 Fully-induced control system

Fully-induced control means that all streams in the intersections are monitored. In other words, there are sensors in all traffic directions as Figure 1.5. Therefore, this mode of control system is very suitable for intersections where traffic volumes are largely random and change throughout the day. Therefore, there are several advantages obtained through this control system:

Figure 1.5: A fully-induced system representation

- The traffic delays can be greatly reduced by a more sensible structure to the different volumes of traffic;
- The information obtained by sensors can help the control center to optimize green time allocation;
- Improve the efficiency of vehicles movement.

1.3.4 Disadvantage of convolutional TSCS

However, there are shortcomings in these system methods shown shortly in table 1.1

Non induced	Semi induced	Induced
• It cannot compensate for unexpected changes that occur in the volume of traffic.	• an excessive delay of traffic can be caused by the continuous call from the secondary road	• It takes a lot of detectors, which cost a lot of money.
• it is ineffective to the intersection where the traffic is random.	• some detectors should be installed in the small road, which costs a lot of money and maintenance.	• it takes a long time and a lot of people to maintain this system

Table 1.1: Table showing disadvantages of TCS methods.

1.4 Some existing traffic control systems

The timing of signals at road intersections has an important impact on traffic congestion levels, both at the intersection itself and at surrounding intersections that can be signaled or controlled according to priority. Ensuring that traffic signals are timed effectively is one of the most cost-effective ways to reduce congestion.

There are many well-known existing traffic control systems. In the following, we will provide an overview of each system.

1.4.1 Traffic Network Study Tool (TRANSYT)

TRANSYT[41][45] is a method that has been used in Chile since the early 1980s for determining optimal settings for a constant-time signal that enables known streams to pass through a road network with minimal resistance in order to optimize the traffic flow within an area. The method is performed by a digital computer program. A traffic signal model is used whereby permitting flow interaction between successive sections of the road is customized and the flow control is controlled by signals with a fixed, predetermined time based on prior information. Optimizes saturation, congestion time, fuel consumption, total stops, running cost, total travel, total delay, average delay, and system speed. Allows optimization of both signal offset and green times assignment according to the area.

Networks can have up to 50 intersections.

1.4.2 Microprocessor Optimised Vehicle Actuation (MOVA)

MOVA [46] is a new signal control strategy researched and developed by TRRL for semi-isolated junctions. The data from vehicle detectors on the links are analyzed by an internet-connected microprocessor that implements the mova program; The duration of the green signals is controlled by the logic of delay minimization and, if any branch road becomes oversaturated (congested), through the amplitude maximization process. Mova reduces [39]vehicle delays by approximately 13 percent throughout the working day.

1.4.3 Sydney Coordinated Adaptive Traffic (SCAT)

SCAT[9] is a coordinated adaptive traffic signaling system installed in Sydney, which significantly improves arterial traffic at a low cost, thus enabling optimal use of the arterial road network. An initial trial along an arterial route showed advantages in fsignal time compared to the 35-39 percent fixed time signal format at peak periods. The SCAT is unique in that it is made entirely of computers and secondary road detectors and is fully adaptable to the demands of traffic. Its communications network provides robust and flexible system administration. The benefits are not only in reduced delay, improved flow, and reduced congestion, but also in reduced accidents, reduced use of petroleum resources, reduced air pollution, and improved residential facilities.

1.4.4 Urban Traffic Optimization by Integrated Automation (UTOPIA)

UTOPIA [44] is a high performance adaptive traffic control system designed to optimize traffic flows. It offers a wide range of strategies designed to suit any road network and even unpredictable traffic conditions. In fully adaptive mode, it constantly monitors and predicts traffic conditions and

optimizes the control strategy according to flow efficiency and/or environmental parameters. Utopia tracks public transport via a network of roads and grants special permits at traffic signals. Thus reducing travel time, these systems encourage people to use public transportation and help reduce traffic congestion to a great extent.

1.4.5 Optimised for Adaptive Control Policies (OPAC)

OPAC[20] is an online control algorithm designed to improve the performance of individual traffic signals. It is a building block for demand-responsive control of a distributed signaling system. OPAC-RT is a traffic signal control system that implements real-time OPAC strategy. The system uses traffic data collected from detectors located upstream (400 to 600 feet) from the stop bar on all roads leading to the intersection. The timing of the signals is dynamically optimized in a way that responds to demand using a rolling horizontal chart.

1.5 Intelligent Traffic Signal Control System (ITSCS)

Existing traffic signal control systems still rely heavily on simplified information and rule-based approaches, although we now have richer data, more computing power, and advanced methods to drive the development of intelligent transportation, which is why researchers are considering developing more efficient intelligent traffic system. ITSCS[23] are an innovative approach to real-time traffic signal control, can form a large part of the intelligent transportation system, which is widely used in the traffic system of various developing cities. It combines artificial intelligence and the current traffic signal control. It improves the traffic performance of traffic signals already on the road, improves traffic flow in urban networks and lanes, and reduces waiting times, congestion, short trips, less pollution and happier drivers. A good dynamic system receives real-time information from a software API based on data received from all capturing devices such as cameras, radars and even sensors. According to artificial intelli-

genence It process this information then use dynamic algorithmic signal control methods to create an improvement plan every second to control traffic signal status at intersections as efficiently as possible.

1.6 Conclusion

In this chapter we have represented the Traffic Signal Control System and its types, where these systems used a constant time control method in which signal timing plans are switched to run depending on the time of day. The research then led to the development of several traffic-responsive approaches that allow the basic controls-lap time, phase division, and displacement-to vary according to prevailing traffic conditions, with some examples of existing systems focusing on their point of weakness. The objective of this thesis is developing a system based on computer vision to control and optimize road traffic by analyzing images and videos captured in a crowded intersection for detecting and tracking vehicles in order to count them. In next chapter we will introduce computer vision and its approaches, also we will talk about image processing then how to detect vehicles in images.

Chapter 2

Computer Vision

2.1 Introduction

Previously, we talked about the existing traffic signal systems and discussed their types with mentioning some examples of each kind of them, while we aim to create an intelligent system based on computer vision. So, what is meant by it, and how can it be achieved?

Computer Vision (CV) is a field of artificial intelligence (AI) that deals with how machines can interpret and understand the visual world.

The main aim of computer vision is to make a machine see like a human. So that it can see and recognize objects regardless of scene conditions that might be bad weather, darkness or blurriness during capture. When we say machine we mean anything that has computing capabilities as robots, drones, self-driving cars. For humans, seeing is made in which the eye and the brain play a key role. To emulate this system, the machine needs an alternative to each one of the two components. Digital cameras play the role of human eye as they capture images, while computer software and IA algorithms play the role of the brain[43], where they try to analyze and interpret those images because the image to the computer looks like a huge set of integer values. Computer vision algorithms extract a description from the image, which could be an object, a text, a 3D model, etc. Applications that form the backbone of computer vision such as object detection, object recognition, object tracking, etc that we will explain later.

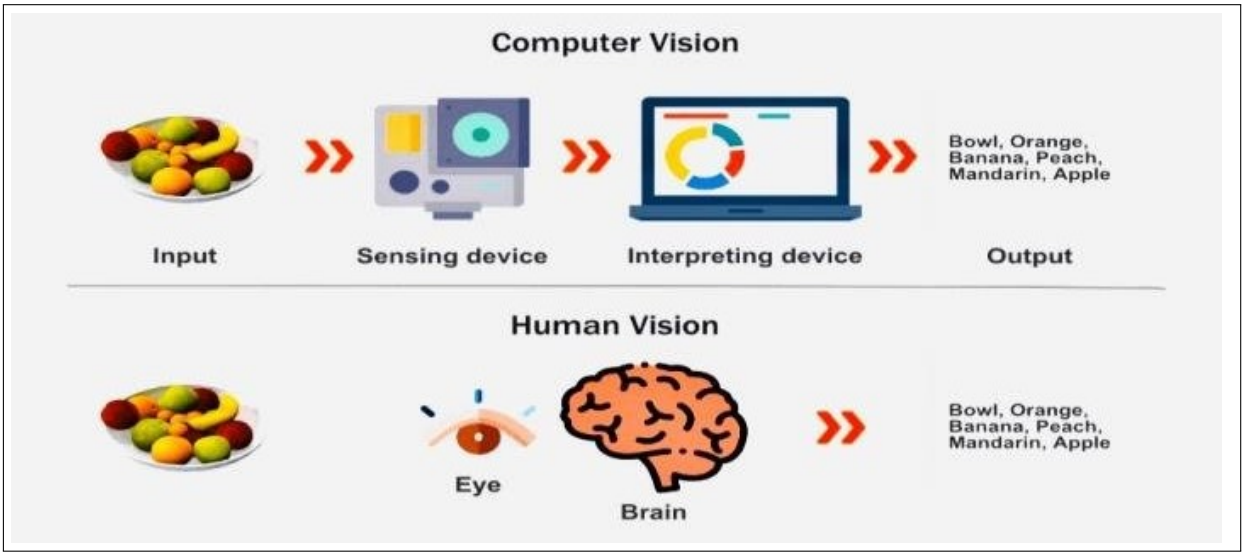


Figure 2.1: Computer Vision VS Human Vision

2.2 History of Computer Vision

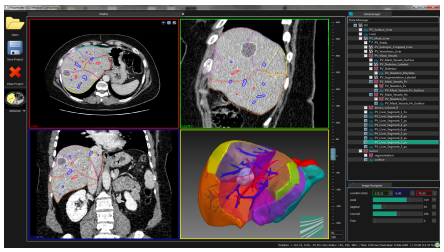
Its commonly accepted that the founder of computer vision is Larry Roberts. In his PHD thesis [5] he discussed the possibility of extracting three-dimensional geometric information from a two-dimensional view, that viewed the project as a springboard for artificial intelligence. In 1966, American computer scientist and co-founder of the Artificial Intelligence Laboratory at MIT Marvin Minsky received a summer scholarship to hire a first-year undergraduate student, Gerald Sussman, to spend the summer connecting a camera to a computer and getting the computer to describe what he saw. Needless to say, ussman did not set a deadline. "Vision turned out to be one of the most difficult and frustrating challenges in artificial intelligence over the next four decades, Sussman chose to never work in vision again." As the Internet became a mainstay, computer scientists had access to more data than ever before. Computing hardware continued to improve as costs fell. Primitive neural networks and algorithms developed in the 1980s and 1990s have improved. Now over half a century ago, the field of artificial intelligence finally had a breakthrough moment

in 2012 (ILSVRC)[4]. It then competed to achieve higher accuracy on several visual recognition tasks. From 2010-2011, the error rate for ILSVRC winners was hovering around 26%. Then, in 2012, a team from the University of Toronto entered a deep neural network called AlexNet that changed the game for AI and computer vision projects. Deep neural networks have revolutionized the field of artificial intelligence. AlexNet had an error rate of 16.4%, and in the following years, ILSRVC's error rates dropped to a few percent; Now, deep neural networks are the gold standard for image recognition tasks. [2]

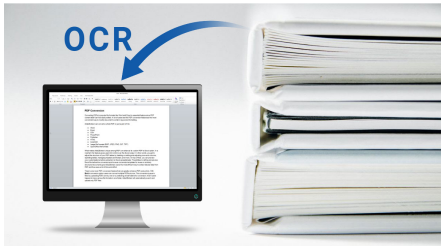
2.3 Computer vision applications

Computer vision being used today in a wide variety of real-world applications [2] like :

(a) Fingerprint recognition

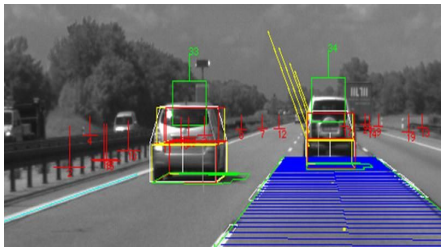


(b) Medical imaging



(c) Optical character recognition (OCR).

(d) 3D model reconstraction (photogrammetry).



(e) Automotive safety.

(f) Surveillance.

Figure 2.2: Some computer vision applications.

(a) Optical Character Recognition (OCR): The technology makes it possible to extract printed or handwritten text from images as reading handwritten postal codes on letters and Automatic Number Plate Recognition (ANPR);

(b) 3D model reconstraction (photogrammetry): 3D modeling is a technique s for producing a 3D digital representation of any object or surface such as Bing Maps;

(c) Medical imaging: Registering pre-operative and intra-operative imagery or performing long-term studies of people's brain morphology as they age;

(d) Automotive safety: Detecting unexpected obstacles such as pedestrians on the street, under conditions where active vision techniques such as radar or lidar do not work well;

(e) Surveillance: Monitoring for intruders, analyzing highway traffic, and monitoring pools for drowning victims;

(f) Fingerprint recognition and biometrics: Refers to the automated method of identifying or confirming the identity of an individual based on the comparison of two fingerprints.

2.4 Image sensing and acquisition

The process of capturing images is similar to that of the human eye. An array of a large number of photosensors serves a function analogous to that of the human retina. The quality of imaging is determined by the number of sensors, sizes and sensing performance. The output of most sensors is a continuous voltage waveform because natural images are generally spatially continuous[19], but for computational purposes, they are often represented by a matrix of discrete samples that can be processed by computers. This is called image digitalization[36] where images are converted to set of values saved in matrix by(M,N)[38], where N and M are the height and the width

of image. This matrix consists of small squares which called pixels, Pixel is a contraction if the term PIcture ELement. It identified by its position coordinates (x,y) in the image.

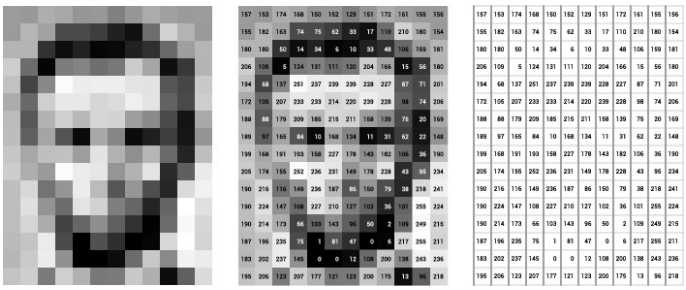


Figure 2.3: Représentation of image in pixels

Though a digital image looks smooth and continuous just like a regular photograph, it's actually composed of millions of tiny pixels as it is shown in fig2.3. We can distinguish two types of images which are colored and grayscale images figure2.4.

(a) Grayscale image (b) colored image

Figure 2.4: grayscale image vs colored image

2.4.1 Grayscale images

One-dimensional digital images are called grayscale images. Typically, individual pixels are encoded using a single numerical value that represents the density of captured brightness[37] as figure 2.4(a) shows. For example, 0 is the darkest (black) and 255 is the brightest (white).

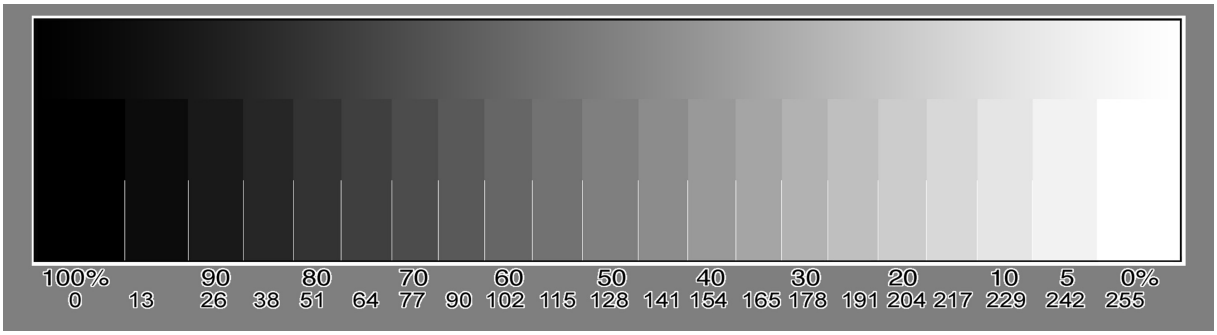


Figure 2.5: grayscale degrees [1]

2.4.2 Colored image

Multi-channel digital images are used to represent color images. The color spaces are based on the composition of colors, which means that the mixing of multi components produces a color figure 2.6. Whereas a pixel is encoded by these numerical values, which is a vector that represents the color at that point [37]. The meaning of these values depends on the type of encoding chosen.

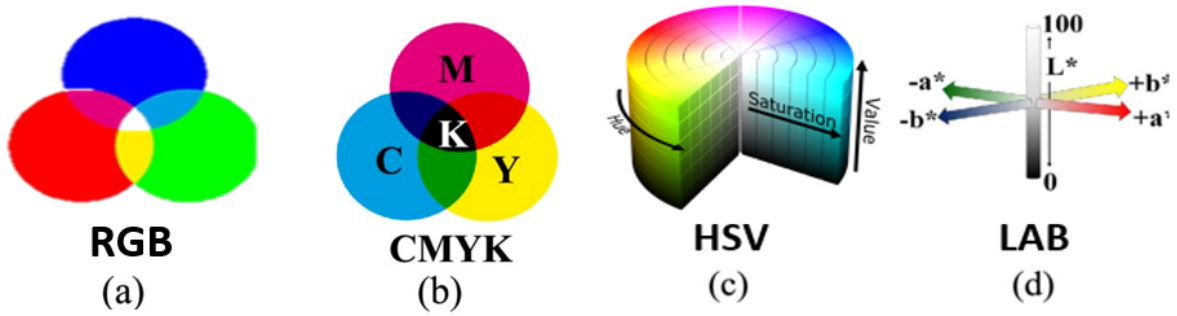


Figure 2.6: Commonly used color spaces.

(a) **RGB Color Space**: The most used to handle digital images is the "Red-Green-Blue" (R, G, B) (RGB) color space,Each pixel in the RGB color space has three dimensions. Examples of RGB colors:

- (R , G , B): model RGB;
- (0 , 0 , 0): black ;
- (255, 0 , 0): red;
- (0 ,255, 0): green;

- (127,127,127): medium grey.

(b) **CMYK Color Space:** The CMYK color space is used in printing, and CMYK refers to the four inks: cyan (C), magenta (M), yellow (Y), and black (BK).

(c) **HSV Color Space**: H (hue)In degrees, S (saturation) specifies the ratio of the purity of selected color to the maximum purity of the color, V (value) represents the luminosity of the color, and overprints four colors with different point area ratios representing rich colors and tones.

(d) **Lab Color Space:** The Lab color space stand for (L) signalness, (a) Red/Green Value, (b) Blue/Yellow Valueis, is larger than that of RGB and CMYK.
Any color in nature can be expressed in the Lab color space.

2.5 Image processing for computer vision

Image processing refers to the application of mathematical functions to images, in order to obtain an improved image or to extract some useful information from them. This is done by inserting an image, and the output may be either an image or properties/features associated with that image figure 2.7. [38] Image processing mainly includes the following three steps:

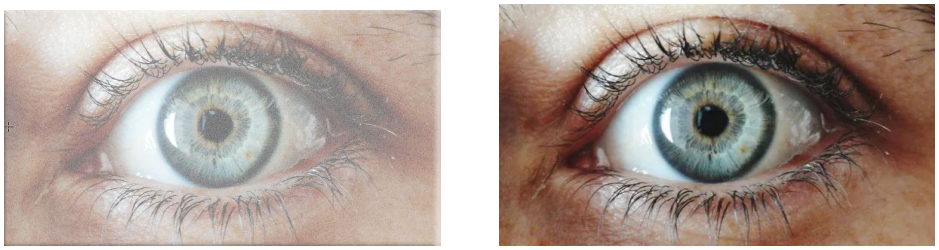
Figure 2.7: Image processing

- Image import via image acquisition tools;
- image analysis and processing;
- The output by which the image can be changed or the report based on image analysis.

It just means that the algorithm performs some transformation on the image, Smoothing, Resizing,Subtraction, sharpening, contrasting, stretching the image, etc.

2.5.1 Image Smoothing

Image smoothing[29][3] is an operation thats used to remove noise, sharpness and clutter in the image to give you much more smoother and blended effect.



(a) Before Smoothing (b) after Smoothing

Figure 2.8: Image Smoothing.

2.5.2 Image subtraction

Image subtraction is generally performed between two images that have significant similarities figure 2.9, in which the numerical value of one pixel or an entire image is subtracted from the other. This is mainly done to detect changes between two images, this change detection can be used to find out if something in the image has moved, and it can also perform background deletion[7].

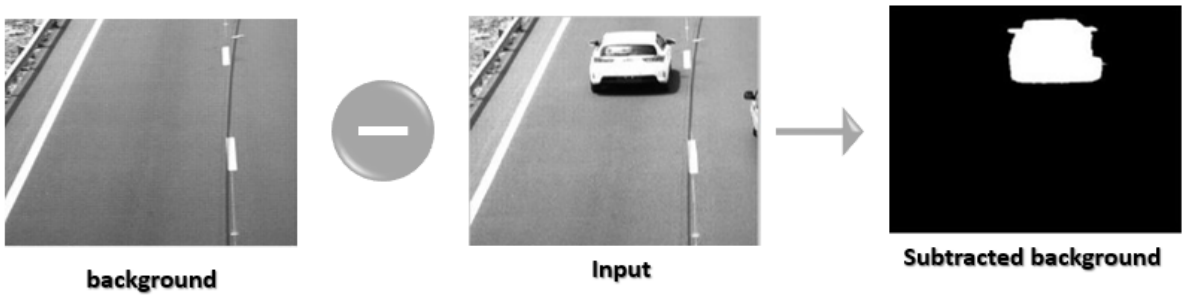


Figure 2.9: image subtraction method

2.5.3 Image resizing

Resizing makes the image larger or smaller figure 2.10, change it to a new width and height to make the image size proportional, and the image becomes smaller or larger without cropping anything, which usually affects the file size and image quality. The most common reason for resizing images is to reduce the size of large files[35].

Figure 2.10: Image Resizing.

2.6 Object detection

Object detection is one of the most active areas of research in the field of computer vision (CV). It includes the classification and location of each object in the image (object localization), to classify and locate every single object within the image. Assign a class to every object and draw a bounding box around it figure 2.11[9], such as humans, animals, cars, or buildings, etc. It is associated with many applications, including image classification, human behavior analysis, face recognition and autonomous driving, etc...

Before going into the details of object detection, the difference between these terms: object segmentation, object detection, object classification

should be clarified, as shown in table 5.2

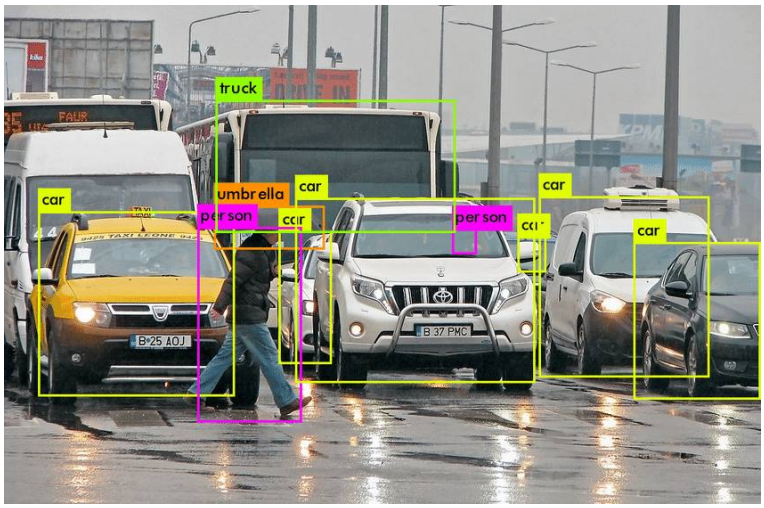


Figure 2.11: Object Detection.

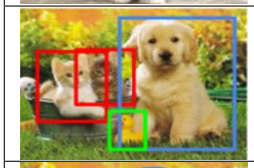
	**Object Classification**	It can find out what the object is in the picture, but not the exact location
	**Object Detection**	Find out the location information of the objects and put a box surrounding this element
	**Object Segmentation**	termed as categorizing each pixel value of an image to a particular class, Give to any objects a clear outline

Table 2.1: The difference between object segmentation, object detection, object classification

2.6.1 Object Detection algorithms

There are many ancient methods of detecting objects in signel image, and we mention the following:

Viole-Jones algortihm

Paul Viola and Michael Jones came up with the idea of Haar Cascades[12], one of the oldest technique used for detection working on grayscale image,

the algorithm looks at many smaller subregions and tries to find a face by looking for specific features in each subregion figure 2.12. It needs to check different positions and scales because an image can contain many faces of various sizes. It distinguishs the black and white color difference, and the feature values of the Haar features are determined by the pixel values of the rectangle.

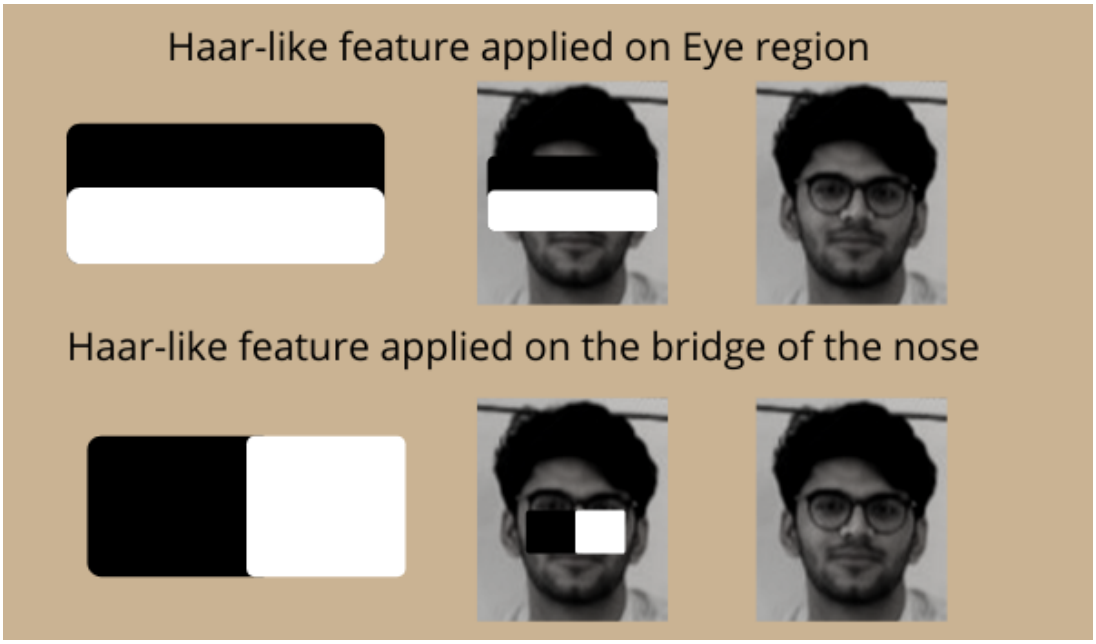


Figure 2.12: viola-jones algorithm.

SIFT algorithm

Scale-Invariant Feature Transform, first introduced in 2004 by D. Lowe. The main idea of the algorithm is that in an image, a neighboring pixel describes what is next to it, in order to identify the nearest neighbors for a better fit image, Lowe's goal is to expand a constant interest factor, rotation, etc. Then, when trying to locate an object in it, the description extracted from the training image can be used to identify the object, keeping the features extracted from the training image recognized even if the image scale, noise, and signaling changes. These points are usually located in high-contrast areas of the image[22].

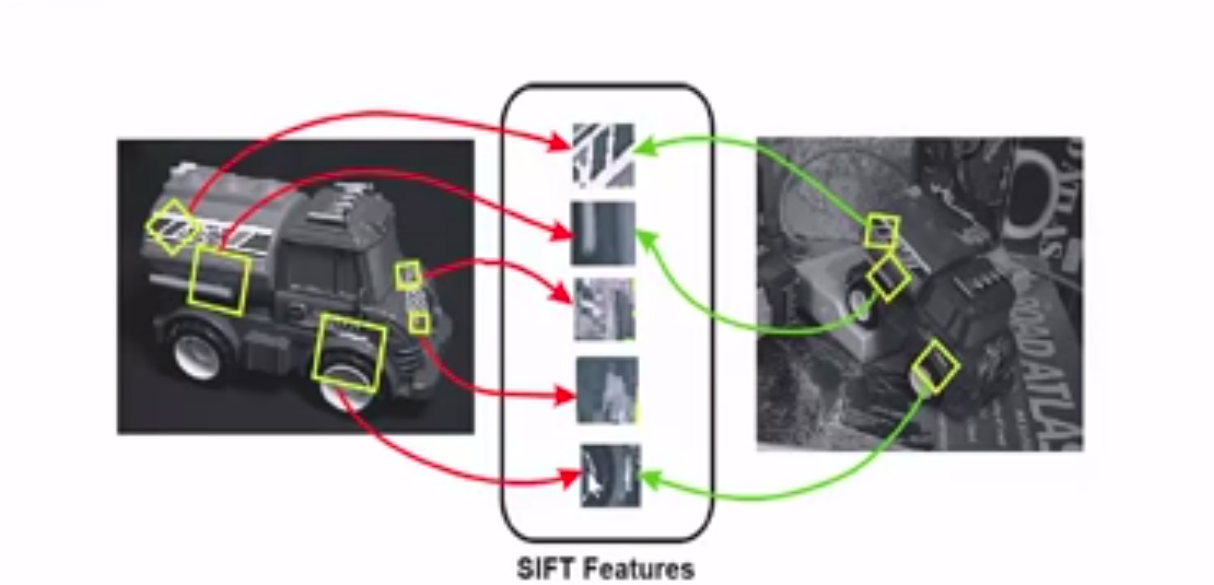


Figure 2.13: SIFT algorithm.

2.7 Conclusion

In this chapter, we presented a definition of computer vision and how images are read and processed, with a mention of some color spaces and some processing methods such as smoothing, subtraction, resizing. We focused on the process of detecting objects in images to extract important informations and work with it, and we explained some of its techniques. This is what we have built in our project to count the number of vehicles in a captured image of the crossroad in order to optimize the time of the traffic signal system, which we will explain in detail in the following chapter.

Chapter 3

State of the art of object detection and vehicle counting

3.1 Introduction

In recent years, artificial intelligence (AI) systems have had a significant impact on human life as they can process large amounts of diverse data and make decisions that help solve many real-world problems to create a better world by achieving common goals that benefit societies and individuals, such as mobility In cities by improving traffic light systems and thus trying to eliminate pollution and improve the urban environment and the quality of life of citizens by monitoring vehicle traffic, especially at intersections.

Vehicle counting plays an important role in smart city transportation systems, where reliable and efficient deep learning algorithms can use cameras to detect,classify and count vehicles in real time. Images are the best sensing data for analyzing and evaluating the flow of vehicles in crowded areas. Current techniques for counting vehicles in images rely on large amounts of annotated data, which requires providing visual data through stationary city cameras, while artificial intelligence systems extract relevant information from this deluge of data using deep learning algorithms.

Transportation is one of the biggest sectors that can benefit from actionable insights from data captured by cameras, as computer vision and deep learning have shown promising results in achieving hands-on deployment at scale. In this section, we will cover different vehicle counting

approaches and compare their effectiveness in practice.

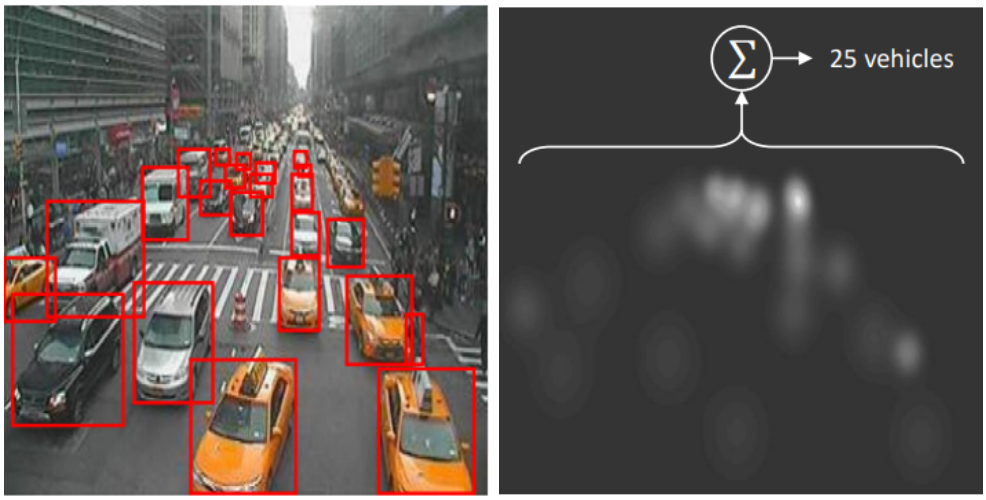
3.2 Vehicle counting

Counting[26] is an estimate of the number of objects instances in still images or video frames. ***Vehicle counting*** is to identify and count different types of vehicles in a single video frame or an image using computer vision and deep learning algorithms.

Vehicle counting approaches can be divided into two main categories[11]:

a. **Detection-based methods:** Which attempt to identify and localize instances (vehicles) in an image regardless of their identity (car, truck, bus,etc.) figure 3.4(a). then count the detected vehicles.

b. **Density-based methods:** [15]Which uses density estimation algorithms to estimate a density map from an image, then to regress the number of vehicles. where the final number is given by summing all values of pixels figure 3.4(b).



(a) detection based method (b) Density-based method

Figure 3.1: categories of vehicle Counting methods

Due to the recent progress in computer vision boosted by deep learning methods for object detection, more studies turn to the detection-based strategy.

3.2.1 Detection based method

Vehicle detection is the first step in the vehicle counting process, which means locating the vehicle and surrounding it with a bounding box that has 4 properties: width, height, and coordinates (x, y) as figure 3.2.

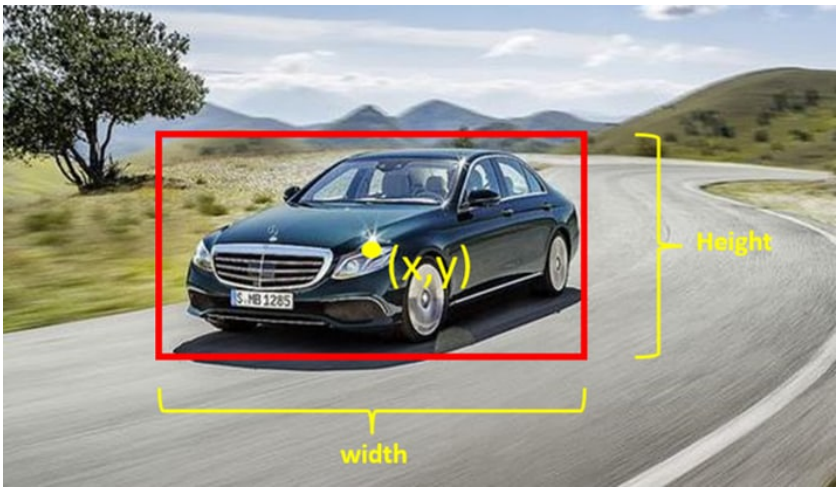


Figure 3.2: Representation of bounding box

The goal of this detection technique is to determine where vehicles are located in a given image, that is called object localisation and which category each object belongs to, that is called object classification. There are several ways to do this detection process as shown in Figure 3.3

feature based vehicle detection

Previously, the detection of objects involved the use of classic algorithms, such as those supported in OpenCV[8]. In addition to background subtraction, most object detection algorithms are built based on manual features such as HOG [42] and SIFT[22], and then use the extracted features to train Naive Bayes, SVM[24] and other classifiers to obtain detection results.

But these methods usualy do not perform very well when working under different conditions such as partial occlusion, shadows, and lighting contrast.

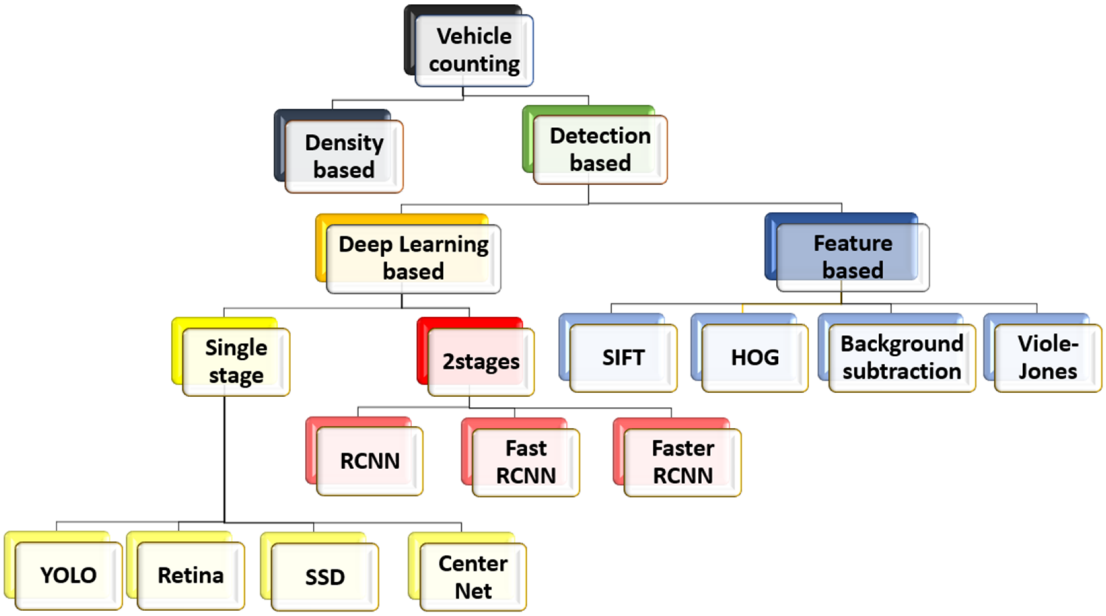


Figure 3.3: Classification of vehicle counting methods

CNN-based vehicle detection

The advent of deep learning has solved many problems in object detection such as partial occlusion, shadows, and lighting contrast. Where has recently taken over the task of object detection by methods based on deep neural network (DNN)[25]. DNN variant of machine learning, it has a high learning capacity when dealing with images. DNN consist of several convolution layers and pooling layers, the last layer is full connection with output(this can be considered as neural network). DNN can automatically learn rich features from trainning dataset, and has achieved excellent performance in many applications such as image classification and object recognition. DNN methods can be divided into two main categories: single-stage and two- stage object detectors

- **two-stages:** It uses a region proposal network to generate regions of interest in the first stage and sends region proposals to object classification and bounding box regression pipelines. Such models achieve the highest accuracy, as Faster R-CNN (Region-based Convolutional

Neural Networks)[40] , but are generally slower.

- **single-stage:** Where the detection process is considered "*complete in one step*", which mean single-stage object detectors make classification and place precise bounding boxes in one pass[18], we can cite as examples YOLO(You Only Look Once)[16], SSD(Singe Shot MultiBox Detector)[30].

3.3 Dataset

Object detection neural networks need to be trained on huge image datasets. As a result, a number of well-known datasets and benchmarks are released by many research institutions, including the datasets of:

Figure 3.4: Example images from Pascal VOC 2007 dataset.

- **PASCAL VOC dataset:**[17] The PASCAL Visual Object Classes (VOC) dataset contains 20 object categories including vehicles, household, animals, and other. Each image in this dataset has pixel-level segmentation annotations, bounding box annotations, and object class annotations. This dataset has been widely used as a benchmark for object detection, semantic segmentation, and classification tasks.
  The PASCAL VOC dataset is split into three subsets: 1,464 images for training, 1,449 images for validation and a private testing set.

- **MS-COCO dataset:**[28] COCO is a large-scale object detection, segmentation, and captioning dataset. It has several features: Object segmentation, Recognition in context, Over 200000 images of the total 330000 images are labeled, 1.5 million object instances, 80 object categories, 5 captions per image. COCO is often used to benchmark algorithms to compare the performance of real-time object detection. The format of the COCO dataset is automatically interpreted by advanced neural network libraries. The COCO dataset classes for object detection and tracking include the following pre-trained 80 objects:

- **ImageNet dataset :** [14]A large image database designed for use in visual object recognition researches, organized according to the WordNet hierarchy in which each node of the hierarchy is depicted by hundreds and thousands of images used in image classification and object detection. It contains 14,197,122 annotated images with more than 20,000 categories.

3.4 Vehicle detection related work

While working on this project, we came across several papers on the topics of traffic systems, traffic analysis, and traffic optimization. We have studied this research and integrated some ideas from its results considering the problems they faced. The works most similar to our research work is the AI city Challenges.

In this section, we briefly describe four different approaches in the AI city Challenge in detection based vehicle counting.

3.4.1 Vehicle Counting Approach Based on Deep Neural Networks

1. Vehicles are **detected** every 15 frames by RCNN (Regional Convolutional Neural Network); region has most probably objects classify them by CNN;

2. Trajectories(R) are extracted by **tracking** corner points KLT tracker where R has a unique label ;

   Simple KLT Algorithm

   (a) Detect Harris corners in the first frame;

   (b) For each Harris corner compute motion (translation or affine) between consecutive frames;

   (c) Link motion vectors in successive frames to get a track for each Harris point;

   (d) Introduce new Harris points by applying Harris detector at every 15 frames;

   (e) Track new and old Harris points using steps 1-3.

3. Assign R to detected bounding box.

This approach [6] has an acceptable performance but not the best.

3.4.2 Multi-Class Multi-Movement Vehicle Counting Based on CenterTrack

- **As input:** Current frame + previous frame + the heatmap with all of the detected centers of objects from previous frame
- **As output:** Object bounding boxes
- **As algorithm:** Simple greedy algorithm which associates object in consecutive frames based on IoU metric of object box

This approach has 80% of efficiency but it still not suitable in very crowded scene

3.4.3 Robust Movement-Specific Vehicle Counting at Crowded Intersections

In this approach[32] a single video is considered as the input where a list of counting vehicles is the output. For the process of detection RCNN

detector trained by COCO dataset is used with ResNet50 feature extractor. DeepSort algorithm adopts a single hypothesis tracking methodology with recursive Kalman filter and Frame by frame data association; This approach has been classified as top5 in AIcity 2020 by 93% of efficiency.

3.4.4 Robust and Online Vehicle Counting at Crowded Intersections

In this paper[34] Jincheng, Lu Meng Xia work on shape-based-strategy using PPYOLO[33] as detector and DeepSort as the tracking algorithm with some modification such as:

1. Motion prediction by Kalman filter;
2. Feature extraction using color histogram, motion and shape feature;
3. Data association where Matching Cascade algorithm is used to get the tracker ID for each detected vehicle;
4. Hungarien algorithm to match the detections and tracklets based on distance;
5. Mahalanobis Distance Smoothness Method is introduced when Kalman filter is used in the case when a vehicle stops or starts in the intersection where velocity of vehicle changes sharply.

3.5 Conclusion

In this chapter, we explained the process of vehicle counting and mentioned its methods, focusing on those based on detection that is divided into feature-based and the deep learning methods that we focus heavily on in our project. In the next chapter we will present our proposal vehicles counting based solution.

Chapter 4

Vehicle counting

4.1 Introduction

Traffic congestion may lead to huge social and economic losses, One of the most important factors of these traffic congestion is poor traffic control at intersections, where drivers have to wait for red lights most of the time, even if the road next to them is empty. Therefor counting vehicles is an important step in improving traffic systems. In this chapter, we propose a computer vision and deep learning based solution to solve the problem of vehicle counting.

4.2 System overview

In our attempt to find a solution to the problem of vehicle counting, we propose a computer vision based solution outlined in figure 4.1.

Our method starts with capturing an image from camera installed on intersection, to be able to count vehicles they need to be detected first.

Vehicle detection is a special case of the general computer vision problem of object detection. In the previous chapter, we mentioned some classical object detection techniques such as background subtraction and Histogram of gradients (HOG)[42]. Recently, DNN[25] based methods such as R-CNN [10] and Faster-CNN[40] have achieved good results in computer vision tasks and specially object detection.

YOLO[16] is one of state-of-the-art object detection CNN models, and

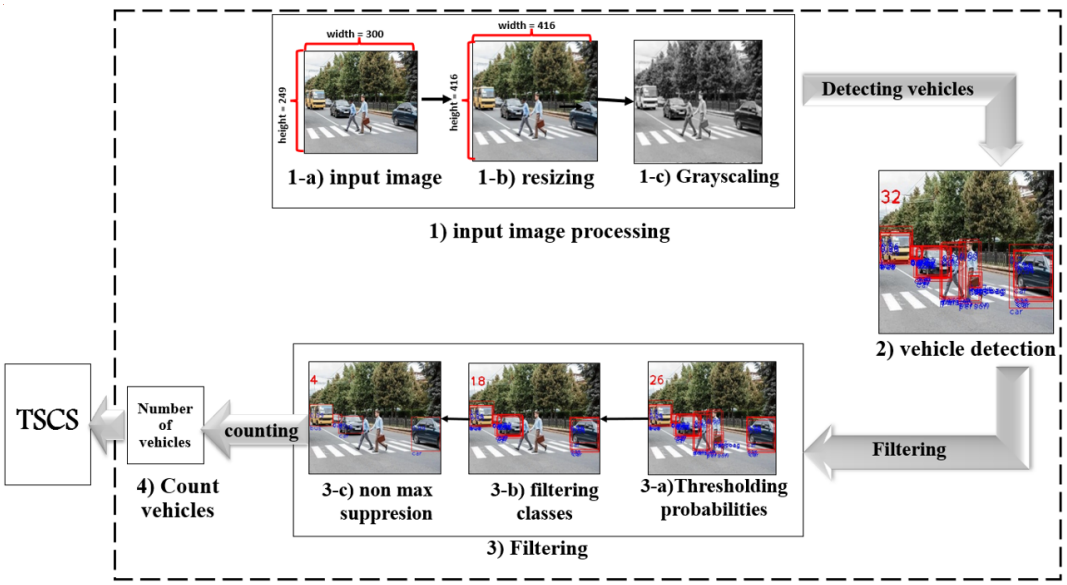


Figure 4.1: Overview of our counting vehicles proposed approach

chosed it due to its superior performance compared to other object detection techniques in terms of accuracy and speed figure 4.2.

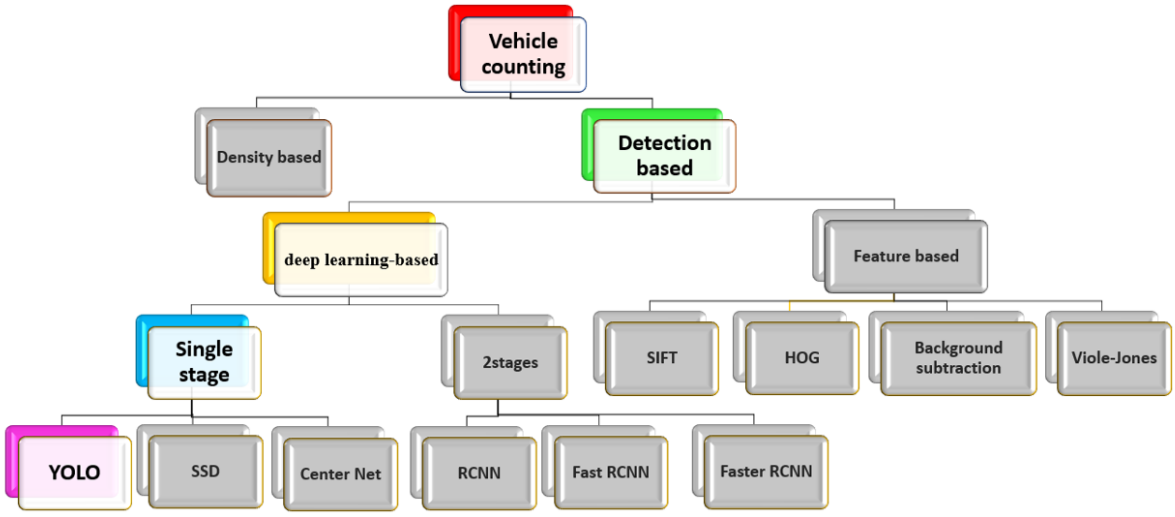


Figure 4.2: Our chosed vehicle detection method

We build our solution on top of the YOLO object detector. The model expects input images in grayscale format with a square shape of 416x416 pixels. Yolo can detect many object categories(person,tree, etc.). For each detected object the model returns the object bounding box and a detection confidence.

Since we are only interested in counting vehicles, we filter objects that do not belong to vehicle's categories(truck, bus, car, bicycle). To avoid er-

rors in the vehicle counting, we first ignore low confidence detections, then we ignore tiny and overlapping bounding boxes using the non max suppression algorithm[21], finally the count of vehicles is given by the number of remaining bounding boxes. Algorithm 1 outlines the main processing steps of our vehicle counting method.

Algorithm 1: Counting vehicle

Input: Image I; confidence threshold T_c; NMS threshold T_{NMS};
Output: number of vehicles N_v.
$L \leftarrow \emptyset$;
$I' \leftarrow Resize(I, 416, 416)$;
$I'' \leftarrow Convert\_to\_grayscale(I')$;
$M \leftarrow YOLO(I'')$;
`/* Applay YOLO object detection to input image */`
foreach *row* R *in* M **do**
    $class\_id \leftarrow \underset{i \in [5,85]}{\text{argmax}}(R_i)$
    if $class\_id \in V$ *and* $R_{class\_id>=T_c}$ **then**
        add $(R_0, R_1, R_2, R_3, R_4, R_{class\_id}, class\_id)$ to L;
    end
end
`/* Non max suppression */`
Sort(L) by descending confidence values
foreach *bounding box* p *in* L **do**
    foreach *bounding box* q *in* $(L - \{p\})$ **do**
        if $p[7] == q[7]$ **then**
            $r \leftarrow Intersectionoverunion(p, q)$;
            if $r <= T_{NMS}$ **then**
                remove q from L;
            end
        end
    end
end
$N_v \leftarrow N(L)$
Return N_v.

4.3 Proposed approach

Previously, we have introduced briefly the general outline of our approach. In the following sections, we will give fully details about each step and we will use figure 4.3.1-a) as input image to explain in details the rest of the steps and their results.

4.3.1 Input images pre-processing

After collecting the captured images from cameras, the model we adopted accepts only square-shaped images of size 416x416 in gray-scale format. So we will perform two operations in the processing : resizing figure 4.3.1-b), grayscaling figure 4.3.1-c).

Figure 4.3: Image pre-processing grayscaling

4.3.2 YOLO vehicle detection

YOLO is an abbreviation for the term You Only Look Once. It is an convolutional neural network model that detects and recognizes various objects in an image. Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images. YOLO is implemented using the OpenCV deep learning libraries, As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects. This means that prediction in the entire image is done in a single algorithm run.
YOLO algorithm is important because of the following reasons:

- Speed, can process 45 frames per second because it can predict objects in real-time and requires only one forward;

- Accuracy, provides accurate results with minimal detection errors;

- The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection;
- it can detect multiple objects, predict their classes and identify their locations.

In order to know this model more, we briefly present the model details:

- **YOLO's input:** A grayscale image with the square shape (416, 416);
- **YOLO's weights:** We will use YOLO's COCO pretrained weights that can detect objects from the 80 classes that come with the COCO dataset[28];
- **YOLO's architecture** The figure 4.4 shows the architecture diagram[48] of YOLOv3

Layer	Filters size	Repeat	Output size
Image			416 × 416
Conv	32 3 × 3/1	1	416 × 416
Conv	64 3 × 3/2	1	208 × 208
Conv Conv Residual	32 1 × 1/1 64 3 × 3/1	Conv Conv Residual × 1	208 × 208 208 × 208 208 × 208
Conv	128 3 × 3/2	1	104 × 104
Conv Conv Residual	64 1 × 1/1 128 3 × 3/1	Conv Conv Residual × 2	104 × 104 104 × 104 104 × 104
Conv	256 3 × 3/2	1	52 × 52
Conv Conv Residual	128 1 × 1/1 256 3 × 3/1	Conv Conv Residual × 8	52 × 52 52 × 52 52 × 52
Conv	512 3 × 3/2	1	26 × 26
Conv Conv Residual	256 1 × 1/1 512 3 × 3/1	Conv Conv Residual × 8	26 × 26 26 × 26 26 × 26
Conv	1024 3 × 3/2	1	13 × 13
Conv Conv Residual	512 1 × 1/1 1024 3 × 3/1	Conv Conv Residual × 4	13 × 13 13 × 13 13 × 13

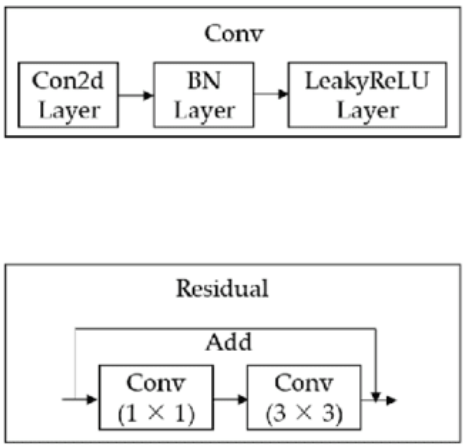


Figure 4.4: Architecture diagram of YOLOv3[48]

- YOLOv3 uses a variant of Darknet, which originally has 53 layer network trained on Imagenet stacked with 53 more layers, giving us a 106 layer fully convolutional;

- The detection is done by applying (1 x 1 x 3) detection kernels on feature maps of three different sizes at three different places in the network (layer 81, layer 94 and layer 106);

- The shape of detection kernel is 1 x 1 x (B x (5 + 85)); Here B is the number of bounding boxes a cell on the feature map can predict;

- YOLO v3 uses binary cross-entropy for calculating the classification loss for each label while object confidence and class predictions are predicted through logistic regression;

- YOLO makes use of only convolutional layers, making it a fully convolutional network (FCN)[47] in YOLOv3;

- Convolution layer is used to convolve multiple filters on the images and produces multiple feature maps;

- Every convolutional layer is followed by a batch Normalizer and a LeakyReLU layers;

- A residual block consists two convolutional layers.

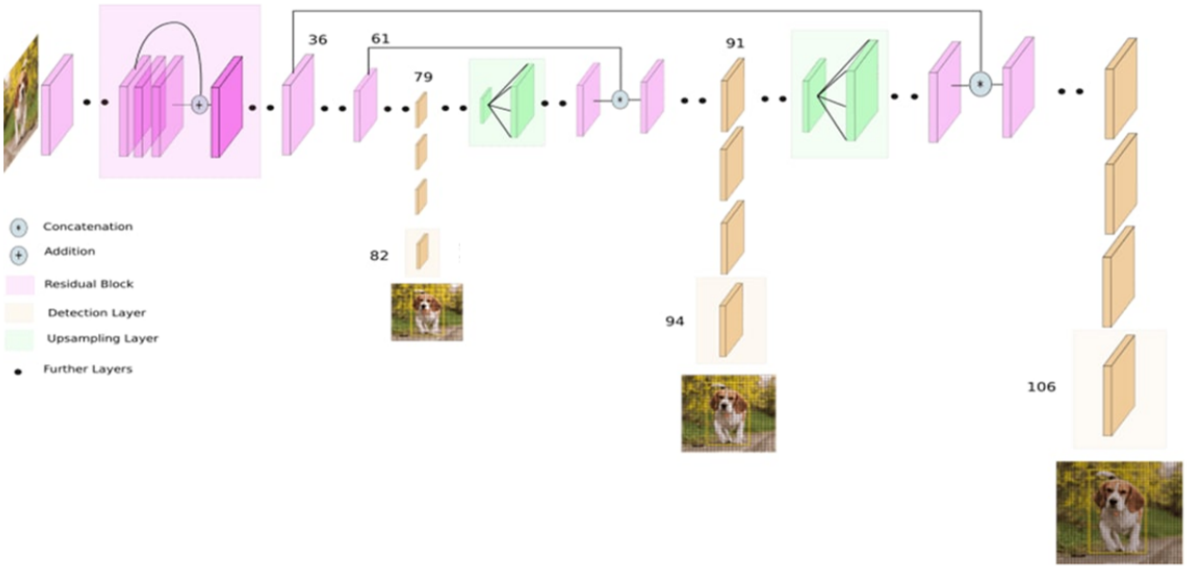


Figure 4.5: YOLO v3 network architecture

- **YOLO's output:** Generally in a sequential CNN network there will be only one output layer at the end. In the YOLO v3 architecture we are using there are multiple output layers giving out predictions as it shown in figure 4.6.

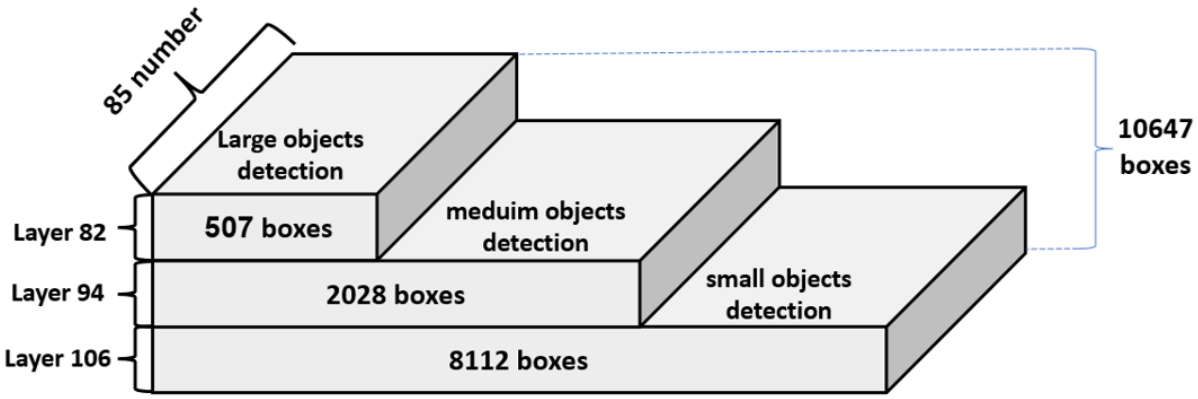


Figure 4.6: Representation of YOLO's output

The output is displayed as matrix. This matrix shows both bounding squares and class probabilities. Each box is represented by 85 numbers figure 4.7.

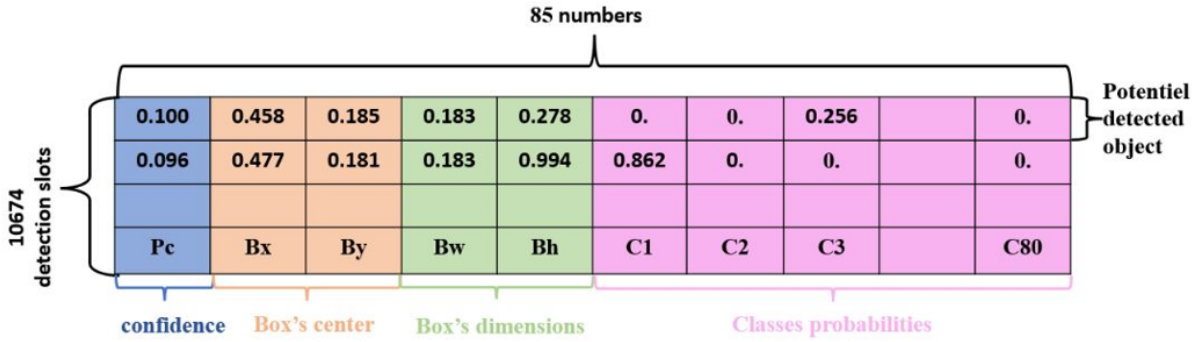


Figure 4.7: Representation of bounding box YOLO's output

Here,

Confidence: Defines whether an object is present in the box or not. It is a probability in [0,1], 0 means no object is detected in this box while 1 means that certainly there is an object;

Box's center and dimensions: Specify the localization of the bounding, see figure 3.2;

Classes probabilities: Probability between 0 and 1 of classes. [c3,c4,c6,c8] represent probability of [car, motorcycle, bus, truck] respectively.

- **Detection using YOLO:** Figure 4.8 ullistrates steps for vehicle detection using YOLOv3

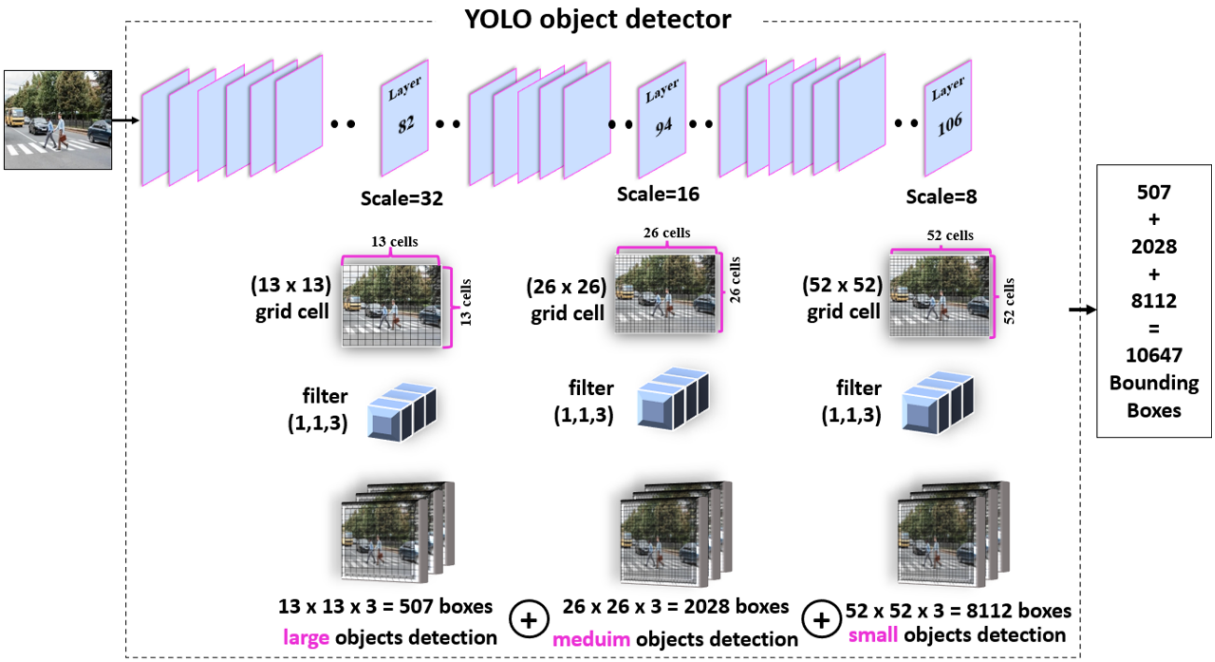


Figure 4.8: object detection by YOLO

The input is an image of shape (416, 416, 3), YOLO passes this image to a convolutional neural network (CNN), this network divides the image into regions and predicts bounding boxes and probabilities for each region. Layers are responsible for the detection are 82, 94 and 106. In each one of three layers the image upsampled by a scale 32, 16 and 8 respectively giving ($\frac{416}{scale}$ x $\frac{416}{scale}$) which means (13x13), (26x26) and (52x52) grid cells respectively. Then detections are made at these layers by kernels (1x1x3) giving (13x13x3), (26x26x3) and (52x52x3) boxes. So this model returns as output
13 x 13 x 3 + 26 x 26 x 3 + 52 x 52 x 3 = 507 + 2028 + 8112 = 10647 boxes that will be filtered to get the accurate boxes.

4.3.3 Filtering

After passing the pre-processed image to the yolo detection model, we get the resulting detection matrix as shown in Figure 4.7. The output matrix encodes all the detected objects with their detection probabilities. The version we used was trained on COCO dataset, and it can detect up to 80 object classes, however since we are interested only in vehicles, the output of the model need to be filtred. Moreover, somtimes the model outputs more than a bounding box in the same area,henece,only one bounding box needs to be preserved using the non max suppresion algorithm, Figure 4.9 is an example of the raw yolo detection output with the above mentioned problemes.

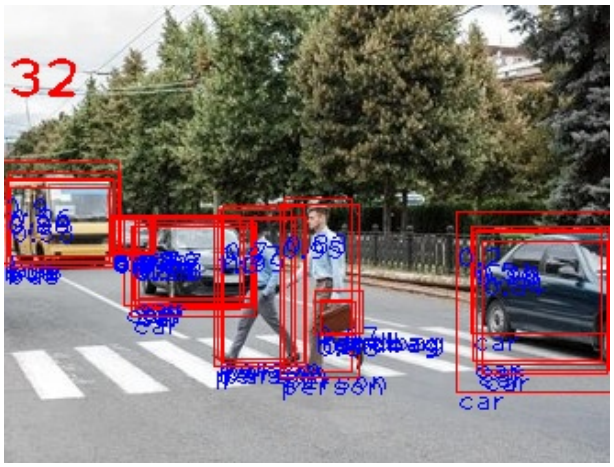


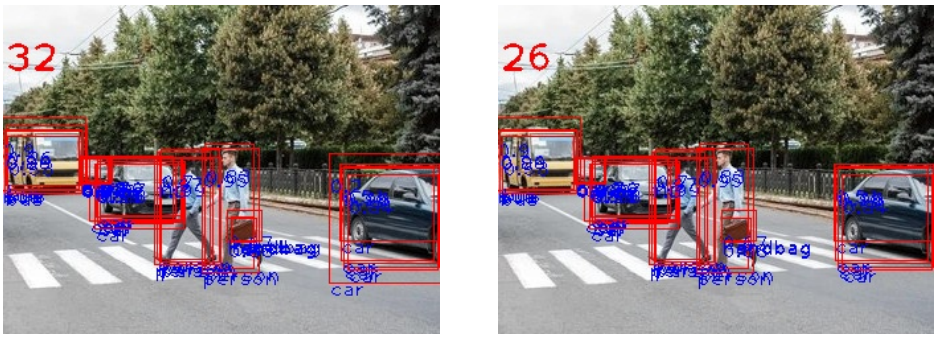
Figure 4.9: A visualisation of bounding boxes

A. **Probability thresholding**

The output of YOLO is a matrix where each row contains the coordinates of the object bounding box with 80 class probabilities. if the probability values are distributed along many classes this means that the model is not confident about the nature of the detected object. In other words, only rows that has a high probability value in one specific class needs to be considered as a valid object detection.

To ignore low confidence object detections, we take each row of the detection matrix and we compare the maximum probability in the row

with a probability thresholding value T_c. if the max probability is grater tha T_c then the row is considered as a valid object detection. In out experiments, we found that a value of 0.5 is a good value for the threshold T_c.



(a) Before (b) After

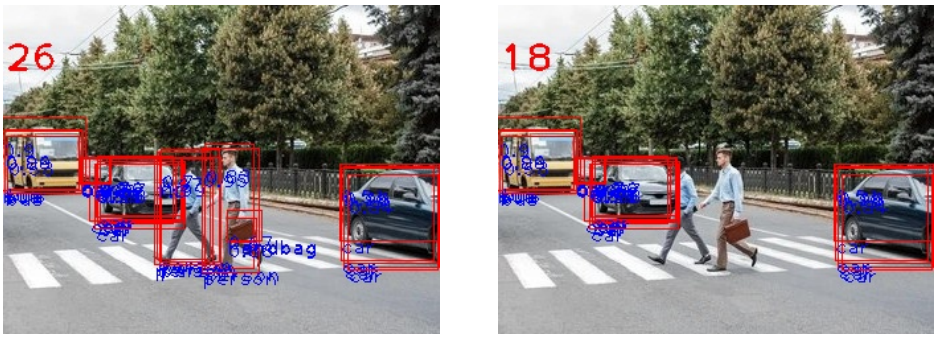
Figure 4.10: Probability thresholding with $T_c = 0.5$

In Figure 4.10.(b) the number of detected objects was reduced after applaying the probability thresholding operation.
shows. Precisely, we said that this model gives 10647 boxes, in our example after thresholding we get just 26 boxes wich represent confidently a present of objects. we plotted only boxes for which the model had assigned a high probability, but this is still too many boxes. We'd like to reduce the algorithm's output to a much smaller number of detected objects.

B. **object class filtreing**

Since YOLO is trained on COCO dataset, it can detect up to 80 different object classes. In our work we are interested in detecting and counting vehicles, so, we need to ignore any object class that dose not belong to the vehicle category V where $V = \{car, truck, bycicle, bus\}$.

For each row in the detection matrix M, we first find the index of the class with the maximum probability. Then, we get the corresponding class from COCO names list. finally, classes that are not in V are ignored. Figure 4.11.(b) shows the result of class filtering operation.



(a) Before (b) After

Figure 4.11: object class filtring

C. Redundant objects removal

As shown in figure 4.11.(b) we can see that multiple bounding boxes are placed arround the same object, this means that the model is detectting the same object multiple times. In our application, this can lead to a wrong ovral estimation of the total number of vehicles. To solve this issue, we need to detects overlapping bounding boxes and remove duplicate ones. this can be accomplished using the non-maximum suppression (NMS) algorithm.

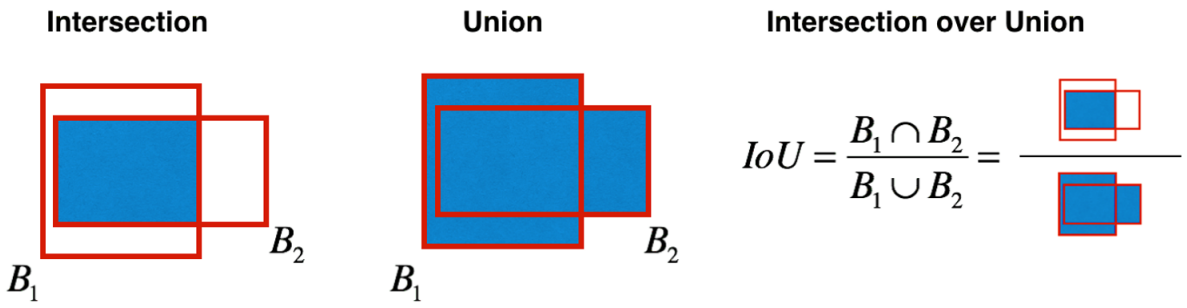
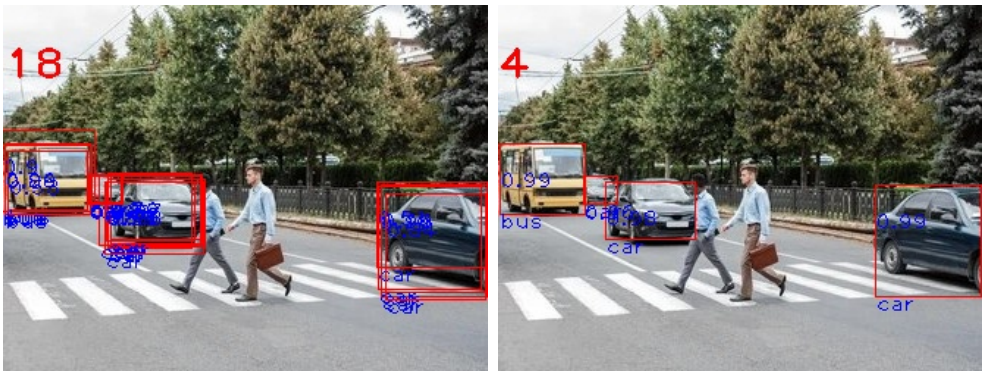


Figure 4.12: Intersection over union ratio computation

The Non Maximum Suppression (NMS) algorithm is a class of algorithms to select one bounding box out of many overlapping ones. NMS is based on the "Intersection over Union" or "IoU" ratio that indecates how much two boxes are overlaping. Figure 4.12 illustrates how "IoU" of two overlapping boxes is computed. Figure 4.13.(b) shows the result of redundant objects removal using the NMS algorithm.



(a) Before (b) After

Figure 4.13: NonMaxSuppression function result

4.3.4 Vehicle counting

As we mentioned before, YOLO's output detection matrix has 10647 rows, each row represents a potential detected object. The number of detected objects is obtained by applying probability thresholding (Section 4.3.3.A) this reduces the number of detected objects to 26 in our example Figure 4.10.(b). Then, we apply object class filtering where non vehicle objects are removed (**person,tree,...etc.**). This reduces the number of objects to 18 in our example Figure 4.11.(b). Then we remove redundant object by applying NMS algorithme (Section 4.3.3.C). This reduces the number of objects to 4 which is the number of vehicles in the input image in our example Figure4.14.

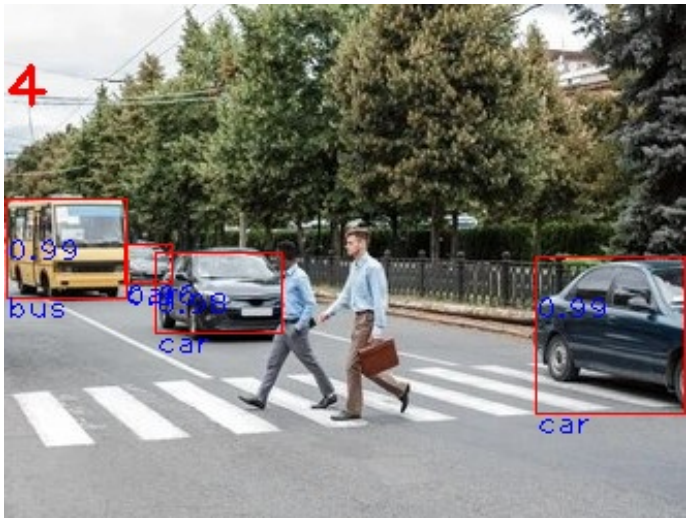


Figure 4.14: Proposed approach's output

4.3.5 Results and discussion

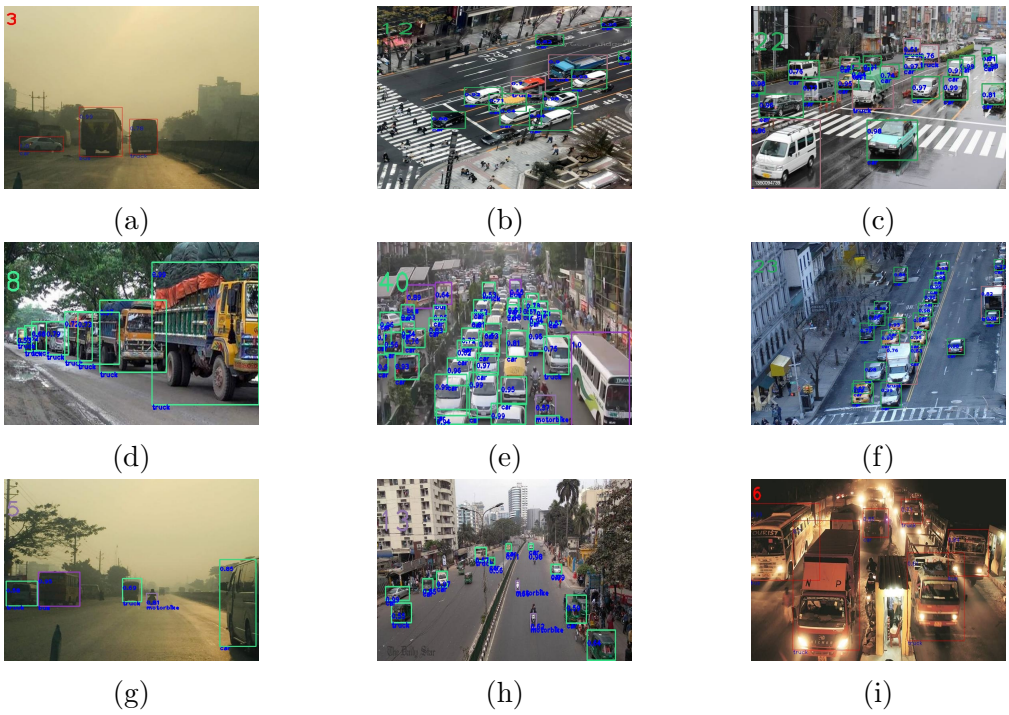


Figure 4.15: some output samples of our vehicle counting method

We evaluated the effectiveness of our proposed method using randomly selected images from the datasets we mentioned before. The images selected, vary on scene complexity, lighting and image capturing angle. Our test set included 3000 images, we used the **mean-precision** metric to evaluate the performance of the proposed approach, **mean-precision** is given by the following formula :

$$precision = \frac{tp}{(tp + fp)} \tag{4.1}$$

Where tp represents the number of true positives (number of correctly detected vehicle), fp is the number of false positives or which represents the number of incorrectly detected vehicles. The Table 4.1 is a sample of mean-precision metric evaluation data we collected. We achieved around 0.8 mean-precision accuracy score witch is quite satisfactory performance. We noticed that in some cases, our method fail to detect and count the correct number of vehicles. This usually happens when in challenging images where cars are so small, images are taken in bad whether, images has

bad illumination, etc.

image	tp	(tp+fp)	fp	precision
1	8	9	1	0.8888
2	9	11	2	0.8181
3	2	2	0	1
4	5	6	1	0.8333
5	5	5	0	1
6	11	14	3	0.7857
7	13	15	2	0.8666
8	6	9	3	0.6666
9	7	7	0	1
10	2	3	1	0.6667

Table 4.1: Sample of mean-precision metric evaluation data

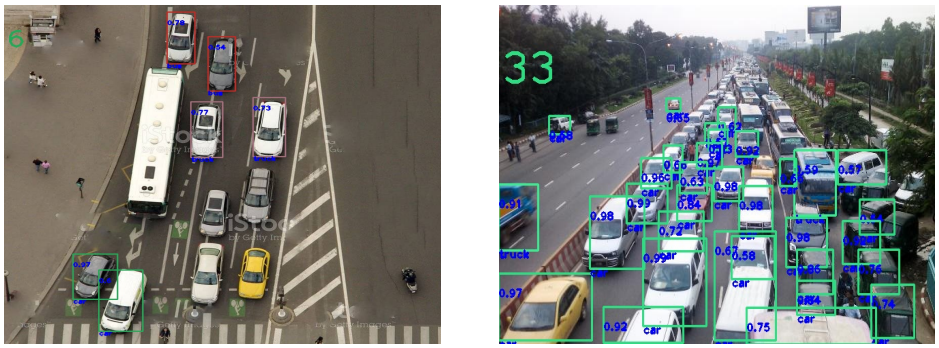
Figure 4.15 represents some samples of our detection and counting approach results. Figure 4.15 shows that in some cases detection and vehicle count are accurate. On the contrary, other images show the less accurate results due to the challenging content of the images and their complexity.

However, in many applications, small error in the final vehicle count is not critical. For example, in the case of traffic optimization system detecting 23 or 26 cars in one direction will lead to the same decision from the system.

4.4 Proposal of a novel vehicle counting dataset

The accuracy of our proposed vehicle counting is tied to the accuracy of the YOLO detection model. Although yolo is the state of the art object detection model, it still suffers from accuracy degradation in the case of very crowded intersection Figure 4.16(a) images, or images taken from Figure 4.16(b).

The limitation of our proposed approach are due to the fact that YOLO is a generic object detector and was not trained exclusively for car detection. One could argue why we didn't trained our own vehicle detection and counting model. The first goal of this work was to propose a new deep learning model dedicated to the task of vehicle counting. However,



(a) (b)

Figure 4.16: our system outputs

to the best of our knowledge there are no publicly available datasets for this specific task. Creating machine learning datasets is not an easy task. Deep learning models are usually trained on datasets with thousands of annotated images. Manually annotating images is labor-intensive task and would take a lot of time and effort. In this work we propose a semi-automated solution for the dataset generation problem.

First we collect a dataset of thousands of images containing vehicles. Then, the approach proposed in Section 4.3 is used to automatically detect and count the vehicles in all the images of the collected dataset. After that, we use a desktop application with a very simple user interface to validate or invalidate the automatic annotation Figure 4.17. The images that where validated by the user are then saved to the final dataset with a JSON file containing the number of vehicles in each image and their bounding boxes.

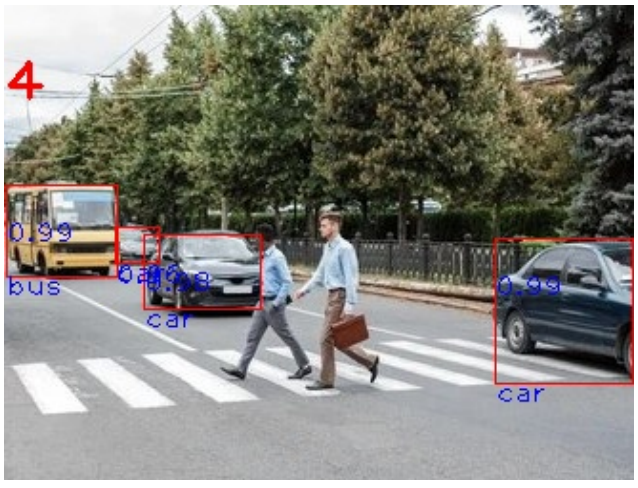


Figure 4.17: dataset's image labeling example

Annotations:
{
image: 1,
number_vehicles: 4,
boxes: {
{257, 122, 72, 76},
{1,94,58,48},
{73, 120, 61, 39},
{58, 116, 23, 19}
} }

Figure 5.19 is a screenshot of our annotation user interface, it was designed to accelerate the annotation process. The user interface provide only two bottons:

- **The button valid** :Used to validate the detection and pass to the next image (the image will be part of the dataset.).
- **The button unvalid** : Is used to invalidate the image due to a detection error (the image will be removed from the dataset)

Figure 4.18: interface for cleaning the dataset

4.5 Perspectives

In this work, we build a vehicle counting solution on top of the YOLO object detection model. We used this solution to create a dedicated vehicle counting dataset. As a followup to this work, we propose the following:

- enhance the dataset creation tool (UI) to add the possibility of correcting the automatic detection and adding manual annotation rather than just refusing the automatic result;

- propose a novel deep learning architecture for vehicle counting that can be trained using out dataset;
- test the proposed approach in real life scenario (use images from real cameras installed in a local city);
- implement the proposed solution as an IOT project.

4.6 Conclusion

In this fourth and final chapter, with the aim of improving traffic light systems, we presented our approach for vehicle detection and counting, we build our solution on top of the YOLO object detection model. We tried to focus on the fundamental aspects of our solutions, hence we intentionally skipped some implementation-related details (development environment, frameworks,code samples).However,if the reader is interested. we provide further implementation details in the supplementary chapter. We provided some experimental results and insights for future work in the Perspectives section.

Supplementary material

5.1 Introduction

In this section, we provide technical implementation details related to our vehicle counting method, first we provide a list of all software,languages and frameworks that have been used in this work. Then, we provide code documentation where we explain the important elements of our implementation code.

5.2 languages, Software and frameworks

- *Python*: (Version: 3.8) Is a high-level, interpreted, general-purpose programming language, easy to learn syntax;
- *Opencv*: (Open Source Computer Vision Library) is an open source computer vision, image processing and machine learning software library.
- *numpy*: is a python library provides support for tons of mathematical and data science functions;
- *Tkinter*: is python library used to build our dataset tool user interface (UI).
- *LATEX*:is a document composition language and system. is was used to write the manuscript of our thesis.
- *VScode*(Visual Studio code): Is a free coding editor that helps for coding quickly. we chosen to use VScode since it has support many lan-

guages that we need, without the need to switching between editors(*Python*, *latex*, *Json*).

- *Git* :A source control system that we used to share our work and collaborate with each others.
- *GitHub* we used Github to host our code and manuscript latex files.

5.3 Code documentation

5.3.1 Code structure

```
.\OutImg
.\InpImg
.\Annotations.json
.\Annotation.py
.\coco.names
.\false_Positives.txt
.\dataset_tool.py
.\initial_annotations.json
.\vehicle_counting.py
.\yolov3.cfg
.\yolov3.weights
```

our project directory contains :

- two folders(InImg, OutImg) for input and output images.
- vehicle_counting.py python script for counting vehicles.
- Annotation.py annotation code.
- dataset_tool.py dataset creation tool
- yolov3.weights Yolo pretrained model weights
- yolov3.cfg Yolo pretrained model configuration file
- coco.names lables for the COCO dataset.

5.3.2 Vehicle detection code

```
import cv2
import numpy as np
import json
classes = []
with open("coco.names", "r") as f:
classes = [line.strip() for line in f.readlines()]
YOLO_cnn= cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
annotations={}
image_annotation={}
annotations_list=[]
```

Loading COCO lables

This YOLO model is trained on COCO dataset, to decode its output correctly, we need to load the CCOCO dataset labels list, this list can be found in the file "coco.names" the file coco.names provides the label name for each object class id example :(car :18; person 16)

```
with open("coco.names", "r") as f:
classes = [line.strip() for line in f.readlines()]
```

Loading and configuring YOLO object detector

To load YOLO pretrained model, we'll take advantage of OpenCV's DNN function called cv2.dnn.readNet. This function requires both a configPath and weightsPath

```
YOLO_cnn= cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
```

Images loading and pre-processing

The input images are loaded from the folder "InImg" for batch processing

```
image_names= os.listdir("InImg")
for img_name in image_names:
    img = cv2.imread(img_name)
    img=cv2.resize(img,(416,416))
```

Vehicle detection using YOLO

In Opencv, the DNN module requires that the input to the neural network should be in special format called "blob". blob prepares the input image to run through the deep neural network. The function cv.dnn.blobFromImage(image, scale, size, mean, swapBR, crop) transforms the image into a blob,

```
    blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0),
   True, crop=False)
```

It has the following arguments:

- the image to transform;
- the scale factor (1/255 to scale the pixel values to [0..1]);
- the size, here a 416x416 square image;
- the mean value (default=0) used to normalize the input image;
- a boolean that specifies if the image needs to be cropped.

The blob object is given as input to the network:

```
        YOLO_cnn.setInput(blob)
```

From the layers of this model we get our results from the last fully connected layer. This model gives three outs from the three layers (82, 94, 106)
giving:

- 507 (13 x 13 x 3) boxes for large objects,
- 2028 (26 x 26 x 3) boxes for medium objects,
- 8112 (52 x 52 x 3) boxes for small objects.

```
    layer_names = YOLO_cnn.getLayerNames()
    layers = YOLO_cnn.getUnconnectedOutLayers()
    output_layers = [layer_names[i[0] - 1] for i in layers]
    outputs = YOLO_cnn.forward(output_layers)
```

The forward propagation takes about 2 seconds on lenovo (Intel(R) Core(TM) i5-4258U CPU @ 2.40GHz 2.10 GH), RAM 8,00 Go, 64 bits operating system.

Filtering

YOLO model returns as output $507 + 2028 + 8112 = 10647$ boxes that will be filtered to get the accurate boxes. for that some lists must be initialized :

```
boxes = []
confidences = []
class_ids = []
vehicles = [ 'car','truck','motorbike','bus' ]
```

- boxes: Our bounding boxes around the object.
- confidences: The confidence value that YOLO assigns to an object, lower confidence values indicate that the object might not be what the network thinks it is
- classIDs: The detected object's class label.
- vehicles: vehicle classes of interest.

a). **Probability thresholding and object class filtering:**
For each box of YOLO's output, the maximum probability(confidence) is compared with a value T_c. If this confidence is grater than T_c and box's class in vehicles then this box is considered as a valid vehicle detection.
In our experiments, we found that a value of 0.5 is a good value for the threshold.

```
        for out in outs:
          for box in out:
            class_id = np.argmax(box[5:])
            confidence = max(box[5:])
            if confidence>=0.5:
                if (classes[class_id] in vehicles):
                    center_x = int(box[0] * width)
                    center_y = int(box[1] * height)
                    w = int(box[2] * width)
                    h = int(box[3] * height)
                    x = int(center_x - w / 2)
                    y = int(center_y - h / 2)
```

```
                    boxes.append([x, y, w, h])
                    confidences.append(float(confidence))
                    class_ids.append(class_id)
```

b). **Redundant objects removal:**

Applying Non Maximum Suppression (NMS) remove overlapping bounding boxes, keeping only the most confident ones.

```
        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
```

Taking advantage of OpenCV's built-in DNN module implementation of NMS cv2.dnn.NMSBoxes() requires the boxes, confidences, confidence threshold(T_c) and NMS threshold a value in [0..1].
In our experiments, we found that 0.4 is a good value for NMS threshold. cv2.dnn.NMSBoxes() returns indexes of boxes that passes the NMS IoU filter.

```
    detected_vehicle=[]
        for i in range(len(boxes)):
            if i in indexes:
                detected_vehicle.append(boxes)
```

detected vehicles is a list of vehicles detected in the image.

Counting

the number of vehicles is the length of detected vehicles in the input image.

```
        numbervehicle=len(detected_vehicle)
```

Annotation

To generate JSON annotation data, we store the data in a python dictionary then we dump the dictionary to a JSON file.
first, initialization of image annotation and bounding box(bb) to empty.

```
image_annotation={}
bb=[]
```

the image annotation contains three keys: image's name, number of detected vehicle in image, and their corresponding bounding boxes.

```
image_annotation['image']=img_det
image_annotation['number_vehicles']=numbervehicle
for vehicle in detected_vehicle:
    bb.append(vehicle)
image_annotation['boxes']=bb
annotations_list.append(image_annotation)
```

Output

this vehicle counting code gives as output two things:

- a folder(OutImg) contains visualization of the counting process, the number of vehicles end their bounding boxes drawn on the input images.
- a JSON file($initial_annotations.json$) contains annotation data.

```
annotations["annotations"]=annotations_list
json.dump(annotations, false_Positives, indent = 6)
false_Positives.close()
```

5.3.3 Dataset annotation code

Dataset creation UI

Our dataset tool, it was designed to accelerate the annotation process, it provides two buttons and display two images as input an image from "InImg" folder which is the original image and the other image it is the same image but after the detection process from "OutImg" folder. a user just need to put the input images in "InImg" folder and the output image in "OutImg" then execute, then the images are displayed as figure 5.19

if the detection is valid click button ✓if it's not then delete it with the button X at the end we will have our valid images and their annotations (json file)

Figure 5.19: interface for cleaning the dataset

1. **Import packages**

```
from tkinter import *
import tkinter as ttk
import os
import json
from PIL import ImageTk, Image
```

2. **Window** A window is an instance of Tkinter's Tk class. Tk() create a window and assign it to root with size ("1000x520")

```
root = Tk()
root.geometry("1000x520")
```

3. **label**

   a Label is to display images, text, buttons to the window(root). in our case we needed two labels

```
lbl = ttk.Label(root)
lbl2=ttk.Label(root)
```

   (a) **Images** to display the images from the both folders (InImg, OutImg)

- os.listdir() requires the folder path, it consider the images as a list;

- Image.open() open the images, which requires the images list;

- each image from each folder in label(lbl, lbl2)

```
        #images with detected vehicles#
          path = "OutImg"
          list = os.listdir(path)
          img=list[n]
          img=Image.open("OutImg/"+img)
          image = img.resize((500,500))
          image_tk = ImageTk.PhotoImage(image)
          lbl['image'] = image_tk
          lbl.grid(row = 0,column = 0)

          #the input image#
          path_detect = "InImg"
          list_detect = os.listdir(path_detect)
          img_det=list_detect[n]
          img_det=Image.open("InImg/"+img_det)
          img_det = img_det.resize((500,500))
          img_det = ImageTk.PhotoImage(img_det)
          lbl2['image'] = img_det
          lbl2.grid(row = 0,column = 1)
```

(b) **Buttons** A button has many options ttk.Button(container, text, command)

The container: is the parent component on which you place the button;

The text: is the label of the button;

The command: specifies a function that will be called automatically when the button clicked. the next two buttons displayed :

- ✓ button

```
          btn = ttk.Button(root, text="\checkmark",width
  ="71", height=1,bg='#0B289E', fg='white', command=Valid)
          btn.place(x=500,y=501)
```

Button command:

it goes to the next image in the images list then display it from two folders(OutImg, InImg)

```
def Valid():
  global n
  global list
  n=n+1
  global list_detect
  img2=list_detect[n]
  imag=Image.open("InImg/"+img2)
  image2 = imag.resize((500, 500))
  image_tk2 = ImageTk.PhotoImage(image2)
  lbl2['image'] = image_tk2

  img1=list_detect[n]
  imag1=Image.open("OutImg/"+img2)
  image1 = imag1.resize((500, 500))
  image_tk1 = ImageTk.PhotoImage(image1)
  lbl['image'] = image_tk1
```

- X button

```
btn2 = ttk.Button(root, text="X",width="71",
height=1,bg='#97321D',
fg='white', command=Unvalidate)
btn2.place(x=0,y=501)
```

Button command:

Unvalidate image means delete it from the folders and the annotation file.

```
def Unvalidate():
  imag_not=list[n]
  imag_dec=list_detect[n]
  print(imag_dec)
  os.remove(r"OutImg/"+imag_not)
  os.remove(r"InImg/"+imag_dec)
  list_del.append(imag_dec)
```

4. **Output**: for getting the image name and used for annotation file.

```
with open(r'false_Positives.txt', 'a') as fp:
   for item in list_del:
        fp.write("%s\n" % item)
```

Annotation

1. **Import necessary packages**

```
import json
```

2. **Inputs:** this script requires two files as input
   - A JSON file ($initial_annotation.json$) contains the initial annotation after detection process. all annotation will be stored in a list l and initial annotation will be cleared.

```
with open('initial_annotation.json') as json_file:
    data = json.load(json_file)
l=data['annotations']
data.clear()
```

   -A text file (del-lis.txt) as input which contains names of images deleted f from the new dataset

```
f = open("del-lis.txt", "r")
```

3. **cleaning the annotation** for each word form the del-list file find this word in annotation list comparing it with the key "image" and delete it.

```
for line in f:
    for x in line.split():
        i=0
        while i < (len(l)-1):
            y=l[i]['image']
            if(x==y):
                del l[i]
                i=(len(l)-1)
            i=i+1
data['annotation']=l
```

4. **Output:** This annotation.py script returns as output a JSON file (Annotation.json) which is the final labeling of our vehicle detection dataset.

```
        json.dump(data, false_Positives, indent = 6)
        false_Positives.close()
```

```
            "annotations": [
                {
                  "image": "image (1).jpg",
                  "number_vehicles": 11,
                  "boxes": [
                    [48, 236, 33, 48],
                    [304, 269, 57, 64],
                    [178, 326, 39, 35],
                    [186, 172, 26, 20],
                    [135, 177, 26, 20],
                    [193, 179, 23, 16],
                    [59, 182, 23, 27],
                    [201, 191, 26, 18],
                    [101, 196, 25, 26],
                    [128, 238, 27, 21],
                    [92, 250, 31, 24]
                  ]
                },
                {
                  "image": "image (2).jpg",
                  "number_vehicles": 23,
                  \\
                  \\
                  \\


                }
              ]
```

5.4 Further results

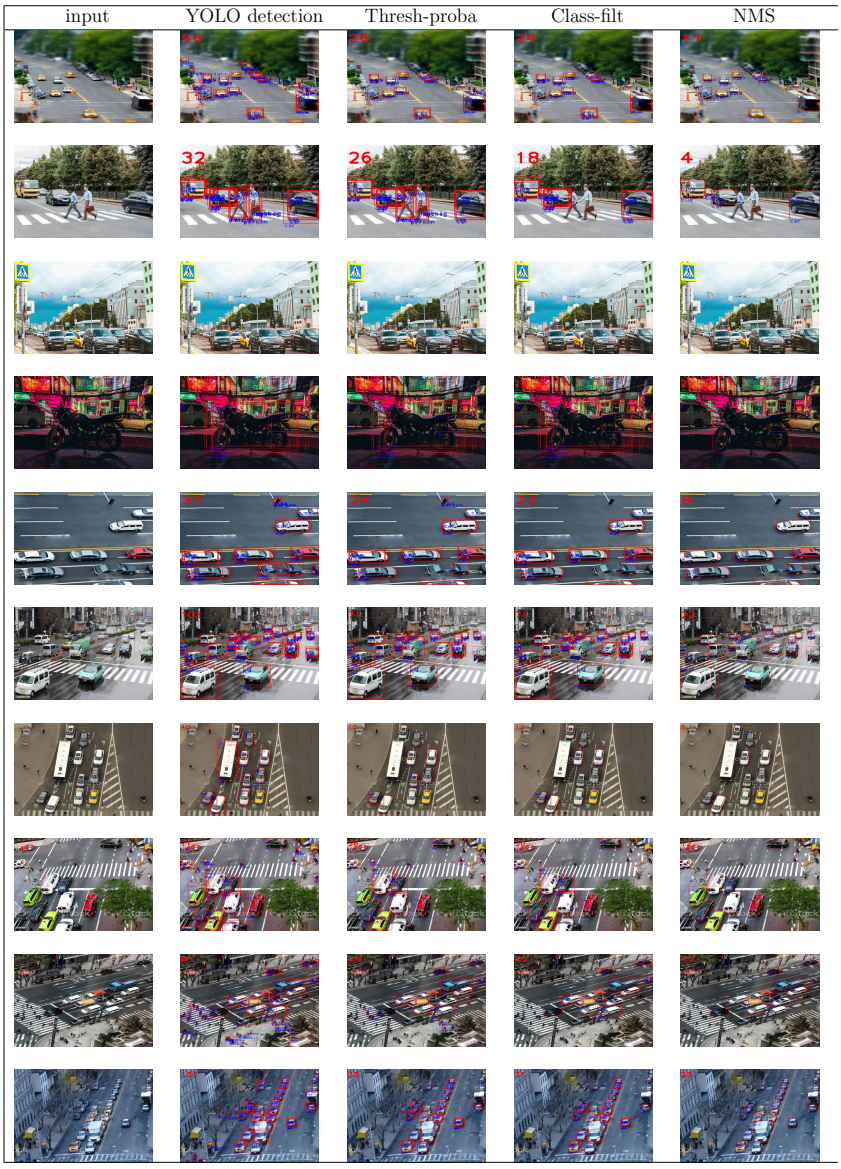


Table 5.2: Vehicle counting results

General conclusion

Traffic congestion may lead to huge social and economic losses, one of the most important factors contributing to traffic congestion is poor traffic control at intersections. Many times, drivers have to wait on red lights, even if the road next to them is empty. Traditional methods for controlling the traffic signal do not prevent the vehicle from getting into the jam,Hence, providing intelligent solutions to Traffic control is an important research area.

One of the fundamental requirements of intelligent traffic control systems is to estimate the number of vehicles in roads and especially at intersections. Sensor based vehicle counting methods have been used to count how many vehicles are present at some point on the road. Recently, computer vision based methods are being proposed as an alternative to sensor based methods. In this work, we proposed a vehicle counting method based on the state-of-the art YOLO object detection neural network model. Our approach is based on filtring the YOLO output layer to extract the number of vehicles in the input image. experimental results shows that the proposed method can achieve hight accuracy scores. However, as many computer vision tasks, the accuracy is sensitive to the quality of the input image(illumination,resolution..etc.).

Since, YOLO is trained on general purpose object detection datasets like COCO its performance on car detection task can not be expected to be optimal. In this work, we proposed a novel car detection and counting dataset with a semi-automatic annotation solution? As a perspective to this work, we aim to use this dataset to train a car detection and counting neural network from scratch to get higher accuracy score.

Bibliography

[1] scale-of-gray, 2014.

[2] How artificial intelligence revolutionized computer vision: A brief history, 2019.

[3] Image smoothing, 2019.

[4] Imagenet large scale visual recognition challenge (ilsvrc), 2019.

[5] Quick history of machine vision, 2020.

[6] Mohamed A Abdelwahab. Accurate vehicle counting approach based on deep neural networks. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 1–5. IEEE, 2019.

[7] C Alard and Robert H Lupton. A method for optimal image subtraction. *The Astrophysical Journal*, 503(1):325, 1998.

[8] Gary Bradski and Adrian Kaehler. Opencv. *Dr. Dobb's journal of software tools*, 3:2, 2000.

[9] Jason Brownlee. *Deep learning for computer vision: image classification, object detection, and face recognition in python.* Machine Learning Mastery, 2019.

[10] Chenyi Chen, Ming-Yu Liu, Oncel Tuzel, and Jianxiong Xiao. R-cnn for small object detection. In *Asian conference on computer vision*, pages 214–230. Springer, 2016.

[11] Luca Ciampi, Carlos Santiago, Joao Paulo Costeira, Claudio Gennaro, and Giuseppe Amato. Unsupervised vehicle counting via multiple camera domain adaptation. *arXiv preprint arXiv:2004.09251*, 2020.

[12] Mehul K Dabhi and Bhavna K Pancholi. Face detection system based on viola-jones algorithm. *International Journal of Science and Research (IJSR)*, 5(4):62–64, 2016.

[13] Luiz Fernando Pinto de Oliveira, Leandro Tiago Manera, and Paulo Denis Garcez Da Luz. Development of a smart traffic light control system with real-time monitoring. *IEEE Internet of Things Journal*, 8(5):3384–3393, 2020.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[15] Petr Dobeš, Jakub Špaňhel, Vojtěch Bartl, Roman Juránek, and Adam Herout. Density-based vehicle counting with unsupervised scale selection. In *2020 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8. IEEE, 2020.

[16] Juan Du. Understanding of object detection based on cnn family and yolo. In *Journal of Physics: Conference Series*, volume 1004, page 012029. IOP Publishing, 2018.

[17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[18] S. Soleimani Fard, A. Amirkhani, and M. R. Mosavi. Retinamhsa: Improving in single-stage detector with self-attention. In *2021 7th International Conference on Signal Processing and Intelligent Systems (ICSPIS)*, pages 1–5, 2021.

[19] Arthur Francisco Araújo Fernandes, João Ricardo Rebouças Dórea, and Guilherme Jordão de Magalhães Rosa. Image analysis and computer vision applications in animal sciences: an overview. *Frontiers in Veterinary Science*, page 800, 2020.

[20] Nathan H Gartner. *OPAC: A demand-responsive strategy for traffic signal control*. Number 906. 1983.

[21] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4507–4515, 2017.

[22] Hailing Huang, Weiqiang Guo, and Yu Zhang. Detection of copy-move forgery in digital images using sift algorithm. In *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, volume 2, pages 272–276. IEEE, 2008.

[23] Somavarapu Jahnavi, G Prasanth, D Priyanka, A Sneheth, and M Navya. Intelligent traffic light management system. In *Proceedings of International Conference on Advances in Computer Engineering and Communication Systems*, pages 489–498. Springer, 2021.

[24] Vikramaditya Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37(2.5):3, 2006.

[25] Qiling Jiang, Liujuan Cao, Ming Cheng, Cheng Wang, and Jonathan Li. Deep neural networks-based vehicle detection in satellite images. In *2015 International Symposium on Bioelectronics and Bioinformatics (ISBB)*, pages 184–187. IEEE, 2015.

[26] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. *Advances in neural information processing systems*, 23, 2010.

[27] Jinjian Li. *Traffic Modeling and Control at Intelligent Intersections: Time Delay and Fuel Consumption Optimization*. PhD thesis, Université Bourgogne Franche-Comté, 2017.

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[29] JG Liu. Smoothing filter-based intensity modulation: A spectral preserve image fusion technique for improving spatial details. *International Journal of Remote Sensing*, 21(18):3461–3472, 2000.

[30] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[31] Zhongji Liu, Wei Zhang, Xu Gao, Hao Meng, Xiao Tan, Xiaoxing Zhu, Zhan Xue, Xiaoqing Ye, Hongwu Zhang, Shilei Wen, et al. Robust movement-specific vehicle counting at crowded intersections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 614–615, 2020.

[32] Zhongji Liu, Wei Zhang, Xu Gao, Hao Meng, Xiao Tan, Xiaoxing Zhu, Zhan Xue, Xiaoqing Ye, Hongwu Zhang, Shilei Wen, et al. Robust movement-specific vehicle counting at crowded intersections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 614–615, 2020.

[33] Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, et al. Pp-yolo: An effective and efficient implementation of object detector. *arXiv preprint arXiv:2007.12099*, 2020.

[34] Jincheng Lu, Meng Xia, Xu Gao, Xipeng Yang, Tianran Tao, Hao Meng, Wei Zhang, Xiao Tan, Yifeng Shi, Guanbin Li, et al. Robust and online vehicle counting at crowded intersections. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4002–4008, 2021.

[35] Arrate Muñoz, Thierry Blu, and Michael Unser. Least-squares image resizing using finite differences. *IEEE Transactions on image processing*, 10(9):1365–1378, 2001.

[36] Arun Netravali. *Digital pictures: representation and compression*. Springer Science & Business Media, 2013.

[37] K Padmavathi and K Thangadurai. Implementation of rgb and grayscale images in plant leaves disease detection–comparative study. *Indian Journal of Science and Technology*, 9(6):1–6, 2016.

[38] Jim R Parker. *Algorithms for image processing and computer vision*. John Wiley & Sons, 2010.

[39] JR Peirce and PJ Webb. Mova control of isolated traffic signals-recent experience. In *Third International Conference on Road Traffic Control, 1990.*, pages 110–113. IET, 1994.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[41] Dennis I Robertson. Transyt: a traffic network study tool. 1969.

[42] Ravi Kumar Satzoda, S Suchitra, and Thambipillai Srikanthan. Robust extraction of lane markings using gradient angle histograms and directional signed edges. In *2012 IEEE Intelligent Vehicles Symposium*, pages 754–759. IEEE, 2012.

[43] Linda G Shapiro, George C Stockman, et al. *Computer vision*, volume 3. Prentice hall New Jersey, 2001.

[44] Arthur G Sims and Kenneth W Dobinson. The sydney coordinated adaptive traffic (scat) system philosophy and benefits. *IEEE Transactions on vehicular technology*, 29(2):130–137, 1980.

[45] Aleksandar Stevanovic, Jelka Stevanovic, Cameron Kergaye, and Peter Martin. Traffic control optimization for multi-modal operations in a large-scale urban network. In *2011 IEEE Forum on Integrated and Sustainable Transportation Systems*, pages 146–151. IEEE, 2011.

[46] RA Vincent and JR Peirce. 'mova': Traffic responsive, self-optimising signal control for isolated intersections. Technical report, 1988.

[47] Zhuoyue Wang. Seg-yolo: Real-time instance segmentation using yolov3 and fully convolutional network, 2019.

[48] Yang Yang and Hongmin Deng. Gc-yolov3: You only look once with global context block. *Electronics*, 9(8):1235, 2020.

[49] Dongbin Zhao, Yujie Dai, and Zhen Zhang. Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):485–494, 2011.