1 2 9 0

UNIVERSIDADE D
COIMBRA

Francisco José Marques Santos

Traffic Speed Estimation, Based on Deep Learning in Real-Time with single Camera and end-to-end 3D Convolution Network

Dissertação no âmbito do Mestrado Integrado em Engenharia Electrotécnica e de Computadores, na especialização de Automação, orientada pelo Professor Doutor Jorge Manuel Moreira de Campos Pereira Batista e apresentada ao Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Ciências e Tecnologias da Universidade de Coimbra.

Outubro de 2021

1 2 9 0

UNIVERSIDADE D
COIMBRA

Traffic Speed Estimation, Based on Deep Learning in Real-Time with single camera and end-to-end 3D Convolution Network

Francisco José Marques Santos

Dissertation for fulfillment of the degree in
Electrical and Computer Engineering Department

Supervisor: Doutor Jorge Manuel Moreira de Campos Pereira Batista

Outubro de 2021

"The harder the conflict, the greater the triumph."

- George Washington

Agradecimentos

Abraçar esta experiência da dissertação é uma tarefa árdua que exige a nossa dedicação, no entanto, esta tarefa revela-se igualmente recompensadora tanto a nível profissional, como pessoal. E, porque esta dissertação marca o fim do meu percurso académico, quero agradecer e lembrar aqueles que representam os pilares que me mantiveram perseverante ao longo deste período. Por este motivo, não podia deixar de agradecer à minha família, em especial aos meus pais e irmão, namorada e amigos de sempre, por todo o apoio e compreensão. Acima de tudo, muito obrigada por acreditarem em mim e pelas vossas palavras de encorajamento, porque um simples "Tu consegues!" fez toda a diferença para mim.

O maior agradecimento não poderia deixar de ser ao Professor Doutor Jorge Batista. Não há gratidão suficiente que chegue para o reconhecimento da sua incansável dedicação e paixão que coloca em tudo o que faz. Um sincero obrigado por me fazer querer ser melhor e por fomentar o desenvolvimento do raciocínio crítico. Agradeço-lhe o rigor, exigência e partilha do seu vasto conhecimento. Obrigado pela dedicação do seu tempo, assim como os conselhos, ajuda, motivação, apesar das dificuldades que foram surgindo ao longo deste processo. Um grande e sincero obrigado!

Aos meus colegas de mestrado e de laboratório do ISR, porque tudo é mais fácil de ultrapassar quando é vivido e com boa disposição e entreajuda. Em especial aos" primos do terceiro esquerdo". A todos vocês pelos almoços, jantares e madrugadas que passamos juntos e acima de tudo pelo vosso companheirismo que irei levar para a vida! Sem dúvida que guardarei na memória o tempo que passámos juntos! Sem vocês, nada teria sido possível. Esta tese é também para vós, pois, sem vocês esta dissertação não seria possível.

A todos, muito Obrigado!

Abstract

The last few years have brought a growth in the usage of technologies like Deep Learning and Convolution Neural Networks. This is possible due to advances in hardware and related areas. In reality, the technology that exists today allows anyone to work with neural networks with low ranked Graphics Processing Unit (GPU).

In this work, the Convolutional Neural Network (CNN) composition and methods utilised to tackle the overfitting problems in the Neural Network (NN) will be introduced. To give an understanding of the results obtained, the Multiple Object Tracking (MOT) metrics used in this work were also presented.

Two different networks were exploited, more specifically Tracking by detection and End-to-end Network. Both were used to get speed estimation of vehicles. Concerning NN, the main objective was to detect objects in an image to feed a tracker. The calibration parameters were obtained through an automatic calibration method developed in ISR. On this system, the mean error achieved for the speed estimation was 3.64km/h.

In the End-to-End network, the Deep learning component is even more notorious. This network estimates the average speed of vehicles around a selected region of interest with few frames in a video without requiring camera calibration and vehicle tracking from previous methods. This network has achieved a mean speed estimation error between 3km/h and 5km/h.

This project was in the interest of Brisa, a toll collection service, aiming at the use of surveillance cameras capable of monitoring traffic. At the same time detecting speed through neural networks was achieved without the need to give pre-calculated information to the network.

Keywords

Deep Learning, Convolutional Neural Networks, Computer Vision, Vehicle Recognition, End-to-end Network

Resumo

Nos últimos anos, observou-se um aumento no uso de tecnologias tais como, *Deep Learning* e as CNN. Isto foi possivel devido ao avanço no hardware necessário ao uso destas tecnologias e nas áreas relacionadas. Hoje em dia, a tecnologia existente permite que qualquer pessoa possa utilizar as redes neuronais com um GPU médio.

Neste trabalho é apresentada uma introdução ás CNN, assim como, a sua composição e métodos utilizados para abordar o problema de *overfitting*. De forma a facilitar a compreensão dos resultados obtidos as MOT métricas utilizadas neste trabalho foram também apresentadas.

Exploraram-se duas redes diferentes, mais concretamente, *racking by detection* e *End-to-end Network*. Ambas foram treinadas de forma a estimar a velocidade de veículos. Relativamente à RCNN, o objetivo foi detetar objetos numa imagem, de forma a alimentar um *tracker*. Os parâmetros de calibração necessários para a estimação de velocidade são obtidos através de um método de calibração automático desenvolvido no ISR. Neste sistema, o erro médio da estimação de velocidade foi de 3,64 km/h.

Na rede *End-to-end*, a componente de *Deep learning* é notória, pois, consegue estimar a velocidade sem necessidade de sitemas auxiliares como a calibração e *tracking*. É possivel observar que esta rede tem grandes potencialidades na estimação de velocidades com o mínimo de intervenção. Com a utilização desta rede foi alcançado um erro entre 3 km/h a 5km/h.

Este projeto foi do interesse da Brisa, um serviço de cobrança de portagens, tendo como objectivo utilizar câmaras de vigilância com capacidade de monitorização do transito e ao mesmo tempo detetar velocidades através de redes neuronais, sem a necessidade de fornecer informação calculada a priori à rede.

Palavras Chave

Deep Learning, Convolutional Neural Networks, visão por computador, reconhecimento de veículos, End-to-end Network

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

BIT Brisa Innovation and Technology

CNN Convolutional Neural Network

DCNN Deep Convolution Neural Network

FPN Feature Pyramid Network

fps frames per second

GPU Graphics Processing Unit

GPS Global Positioning System

IDF1 Identification F1 score

IDFN Identification False Negatives

IDFP Identification False Positives

IDTP Identification True Positives

IDR Identification False Recall

IDP Identification Precision

ITMS Intelligent Traffic Monitoring Systems

IoU Intersection Over Union

ISR Instituto de Sistemas e Robótica

JDE Joint Detection and Embedding

KLT Kanade-Lucas-Tomasi

MAE	Mean Absolute Error
MAP	Mean Average Precision
MSE	Mean Square Error
MOT	Multiple Object Tracking
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
NN	Neural Network
NMS	Non-Maximum Suppression
PP	Principal Point
RGB	Red Green Blue
RCNN	Regional Convolution Neural Network
RPN	Regional Proposal Network
SDE	Separate Detection and Embedding
SORT	Simple Online and Realtime Tracking
SSD	Single Shot Detector
VP	Vanishing Point

1 Introduction

Content

Speed estimation has an important role in road safety, specifically on highways, as it provides information that can be used to enforce speed limits and prevent car accidents. Accidents on highways due to speeding can have disastrous consequences for the drivers because of the high speeds involved. Highway cameras are already installed in order to get traffic information in the road for tooling purposes and those same cameras can be used for estimating the speed of passing cars and other traffic analytic tasks. Estimating Speed with already installed cameras is more efficient since there is no need for new infrastructures or new equipment like radars.

Real-time traffic monitoring has been a challenging task over the past decades. The solution for this problem ranges from piezoelectric sensors [3], smartphone GPS, [4], and also the use of video camera, [5], [6], [7], [8], [9], providing a low-cost, non-intrusive monitoring system with good information. Nowadays, with the increase of faster and cheaper hardware, as well as better quality video cameras, Intelligent Traffic Monitoring Systems (ITMS) are a must have solution.

Vehicle speed estimation through a video camera in real-time is currently the subject of an active research (Figure 1.1). In recent years, several methods have been proposed for vehicle speed estimation. Some methods, such as speed estimation through piezoelectric sensors [3] and loop sensors installed on the road, are very accurate but also very intrusive, expensive and not so efficient since the highway already has pre-installed cameras. The same applies to radar, smartphone, Global Positioning System (GPS) [4], infrared systems, etc. These systems are not so intrusive, nonetheless are more expensive and less efficient due to the use of extra sensors besides the cameras that are already installed in the infrastructure.

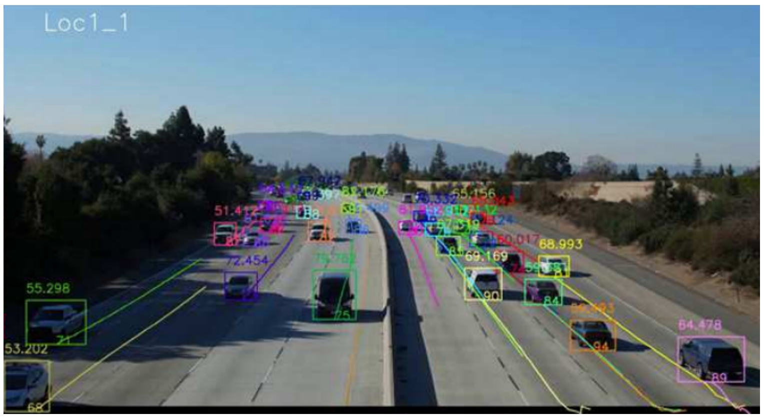


Figure 1.1: Speed estimation with single camera.

Traffic surveillance that makes use of speed estimation can be accomplished through the use of video cameras. Video cameras output data that provides a great quantity of information useful for a multitude of tasks: car classification [10], license plate recognition [11], occupancy detection [12], abnormal events in the road [13] and speed estimation. Therefore, a high quantity of information is being collected nowadays, however it requires a high computational power to be processed. For real-time speed estimation new solutions have been implemented through the use of Deep Affinity Networks [14], Convolutional Neural Network (CNN) on detection phase [15], neural decision tree [8], neural networks and deep learning [16].

Deep Convolution Neural Network (DCNN) achieved incredible success in vehicle detection as well as in the detection of other objects [16]. Recent studies suggest that the use of deep learning to compute feature descriptors and vehicle tracking contribute to better results along with low computational power needed.

Our goal is to implement a system that can give us a real-time solution to vehicle speed estimation. In order for this system to be installed on different camera configurations, the system must adapt to different points of view and different intrinsic camera setups. To accomplish this goal, we will rely on deep learning solutions, through CNN based architecture to implement a real-time speed estimation system where we input video and the system outputs the speed estimation, resulting in an end-to-end system.

1.1 Motivation and Challenges

Speeding is heavily correlated with serious accidents. One way to solve this problem is to understand where high speed is practiced and use methods to reduce it. For this reason, we will need to estimate the cars velocity. The process of estimating velocity in this document will only use video acquired from a single-camera.

Estimating the velocity of vehicles through video can be a challenging task in which some of the problems are: the object occlusion, which can make the tracker loose the object tracking; poor light in the image, gives low contrast and consequently low vehicle detection due to the lost of corners and lines features; low resolution cameras that could not evidence features in the object; camera jitter, which can occur with camera vibration from wind condition e.g.; bad calibration parameters giving wrong spatial object position, etc.

Existing systems shall keep up the evolution of the technology. In that way, Brisa could use their infrastructure to implement the system proposed in this document. This system should be capable of operating independently and run in real-time without loosing performance.

1.2 Objectives

In this dissertation, we are going to use cameras that are already installed in the infrastructure to monitor traffic on highways. The system will output a real-time speed estimation without requiring the incorporation of new equipment, which can be useful to improve the highway monitoring system.

In this work we will use two major detecting and tracking approaches to achieve real-time speed estimation. The initial method computes the speed estimation based tracking by detection method. This initial system has more variables (calibration, detection and tracker) and higher computational requirement.

The second method consists of a deep learning network based on convolution network with low computational requirements. For this second method, to achieve high accuracy it needs to be trained on a very high amount of data. To deal with this high amount of data we plan to use labeled datasets available to the scientific community.

1.3 Contributions

Brisa is the largest private operator of transport infrastructures in Portugal and is nationally and internationally recognized. Brisa operates the key highway concessions in Portugal and is the main shareholder of ”Via Verde”, the most used toll method of payment in Portugal.

To monitor traffic and identify vehicle license plates, Brisa uses appliances that are installed on vehicles and video cameras on highways [17]. This work is within the scope of a R&D project funded by A-to-be, a company owned by Brisa Innovation and Technology (BIT), serving as the group´s international brand responsible for developing and delivering solutions to mobility services.

On this work, the tracking by detection and End-to-end solution for speed estimation are employed and evaluated. The results obtained by this two systems could be of interest to other students and researchers working on similar speed estimation problems.

1.4 Document Overview

This dissertation is organized as follows: firstly, Chapter 2 presents a introduction of the basic information related with this project, as well as relevant approaches used on the project.

In Chapter 3, is presented an explanation of the Multiple Object Tracking (MOT) metrics used to validate results of the the proposed work.

In Chapter 4 presents a discussion of the most important works related with this project, as well as relevant approaches, and a comparison between them.

In Chapter 5 the proposed system is presented, an explanation of each part of the work in detail, clarifying what has been developed, and introducing the main tools used.

In Chapter 6, a simulation study is presented to test different system parameters and initially analyse and discuss system performance under different configurations and at the end tests and the acquired results are discussed in detail.

Finally, in Chapter 7, we make an overview of the developed work and present possible future lines of work.

2

Convolution Neural Network Concept

Content

2.1 Introduction

Recent approaches for object detection use essentially machine learning algorithms. Machine learning performance can increase by collecting more data, use more powerful models and use different techniques to prevent overfitting. Initially datasets were relatively small but with the increase of the use of machine learning, datasets are getting bigger with more general data, increasing the accuracy of systems developed.

Today's Graphics Processing Unit (GPU), combined with a highly optimized convolution implementation, are powerful enough to facilitate training of interestingly-large CNN. knowing this, we will try to implement a solution to a real problem, with an almost infinite number of environments. The amount of dataset used may not be general enough, decreasing precision. That is why it is necessary to have a large amount of data, but also general and balanced data.

To tackle the problem of speed estimation through video from a camera we will use a CNN that is a part of Neural Networks mostly used for image recognition. Images in a CNN will pass through a series of convolution layers with Kernels, Pooling, fully connected layers and apply an activation function. In the next subsections each part of a CNN will be discussed in more detail.

2.1.1 Convolution layer

A convolution layer passes a kernel in the image, creating a new image with enhanced features. Enhancing the image to give defined features is a way to let the network achieve the important features.

On a convolution layer, different types of filters can be used, and those could be adequate depending on the problem. The right use of filters allows the network to successfully capture the spatial dependencies in an image. A convolution layer acquires important features through convolution operations that pass a kernel on the image in a series of convolutions and make calculations between parts of the image and the kernel.

The objective of the kernel is to extract the high-level features such as edges, from the input image. A filter is composed with one or more kernels that slide through the input image.

This sliding of the filter through the input image is defined with 4 hyper-

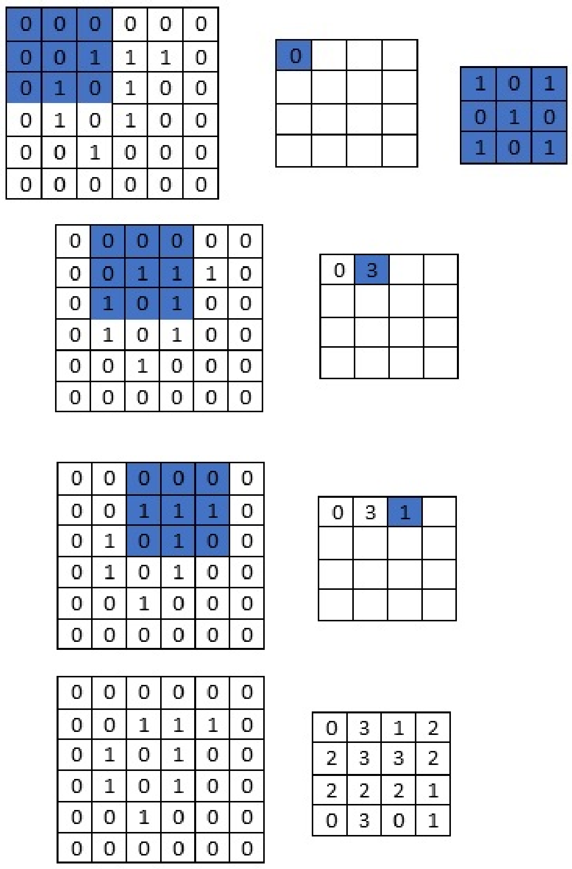


Figure 2.1: Convolution of a 3×3×1 kernel with a 4×4×1, padding of null values, size equals to integer part of kernels half size and stride of 1

parameters:

- Channels, that correspond to the depth of the filter normally 3 for an Red Green Blue (RGB) image;
- Filter size, dimensions of the filter that can change with the interest of extracting features from objects that are closer or distant from the camera;
- Stride, which is the amount of shifting unit from the filter through the input;
- Padding, surrounds the input image with pixels symmetrically allowing it to maintain the same amount of input after convolution.

2.1.2 Pooling Layer

The pooling layer makes a downsampling operation, usually applied after a convolution layer, which does some spatial invariance. The Average pooling gives the average value over a part of the image covered by the kernel, reducing dimensionality. The max-pooling gives the higher number in part of the image where the kernel is applied. The application of Max pooling acts as noise suppression along with dimensional reduction. For this reason, Max pooling is normally used.

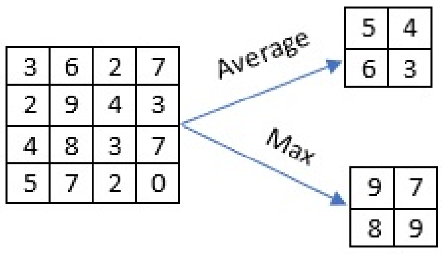


Figure 2.2: Example of max pooling and Average pooling

2.1.3 Activation function

The purpose of an activation function is to add non-linearity to the neural network also decides whether a node should be activated or not(non-linear activation functions). Activation function are normally used between a convolution layer and a pooling layer. After the sums of each node are weighted, the sum is passed through a non-linear function known as an activation function.

The most used function in convolution networks is the ReLU activation function given by Equation (2.1), this function is one of the most used because is computational efficient allowing to train the neural network several times faster without a significant penalty to generalization accuracy. This function will basically output as zero the negative numbers and the positives as their number. It introduces nonlinearities to the decision function and in the overall network without affecting the receptive fields of the convolution layers.

$$g(z) = \max(0, z) \tag{2.1}$$

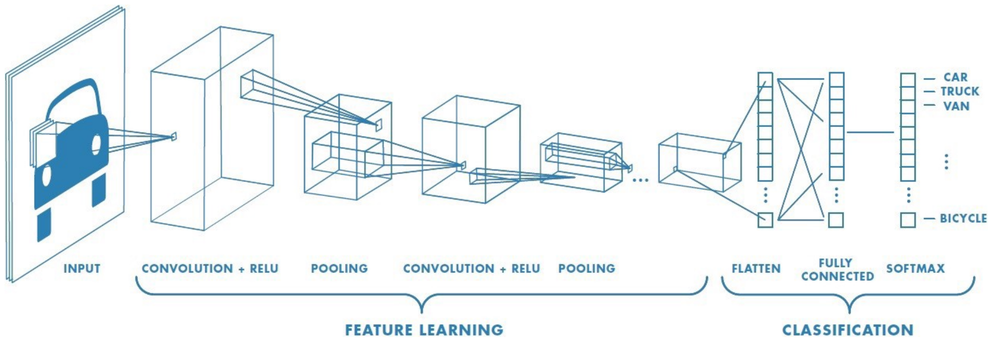


Figure 2.3: An example of a classification CNN.

Other functions can be used like the hyperbolic tangent (2.2) or the sigmoid function (2.3). This last two make the gradient value approaches to zero, which will get the network ceases to learn and suffer from the Vanishing gradient problem in which values away from main range of the function will not have a bigger impact compared to the ones on the main range limit.

Activation function with higher complexity are normally used at the end of the network and just used once. One of the most used activation function at the end of a convolution network is the softmax function (2.4) this function will give a number between 0 and 1 were the higher numbers have an exponential contribution.

$$f(x) = \tanh(x) \tag{2.2}$$

$$\sigma(x) = (1+e^{-x})^{-1} \tag{2.3}$$

$$\sigma(z)_j = \frac{e^{zj}}{\sum_{k=1}^{k} e^{zk}} \tag{2.4}$$

2.1.4 Fully connected layer

The Fully connected layers or feed forward neural network are normally at the end of the network and it is composed by different layers and different connections between each of them. As the name suggests, all neurons in a fully connected layer connect to all the neurons in the previous layer. Each layer is composed with neurons that have different weights. This weights are calculated when the network is training.

The fully connected layer is normally used for classification of the data, after the feature extractions. The use of a fully connected layer is a very powerful tool that have a high training capacity. In contrast, needs a lot of data to train it properly and use an high amount of computational power. With the increase of fully connected layers the density of the network is increased. For this reasons it is used at the end of the network after all the data has been properly transformed reducing it complexity.

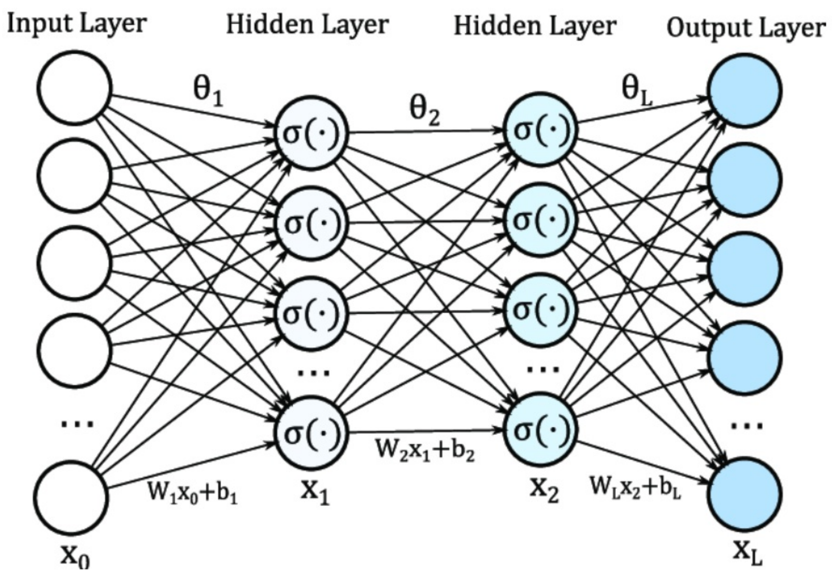


Figure 2.4: Fully Connected Layer representation.

The fully connected layer get as input the last flatten matrix or vector of the neural network and then applies a calculation to the vector $g(Wx+b)$ in which x[p,1] is the input vector, w[p,n] is the weight matrix, b[p,1] is the bias, g is an activation function usually a ReLu, p is number of neurons on the previous layer and n is the number of neurons in the next layer

2.2 Overfitting solutions

Overfitting happens when the model fits too well to the training set. It then becomes difficult for the model to generalize to new examples that were not in the training set.

2.2.1 Dropout

Dropout is a regularization technique that zeros out the activation values of randomly chosen neurons during training. This constraint forces the network to learn more robust features rather than relying on the predictive capability of a small subset of neurons in the network. This significantly reduces overfitting and gives major improvements over other regularization methods. This technique improved the performance of in a wide variety of application including object classification, digit recognition, speech recognition, document classification and analysis of computational biology data as shown in [18].

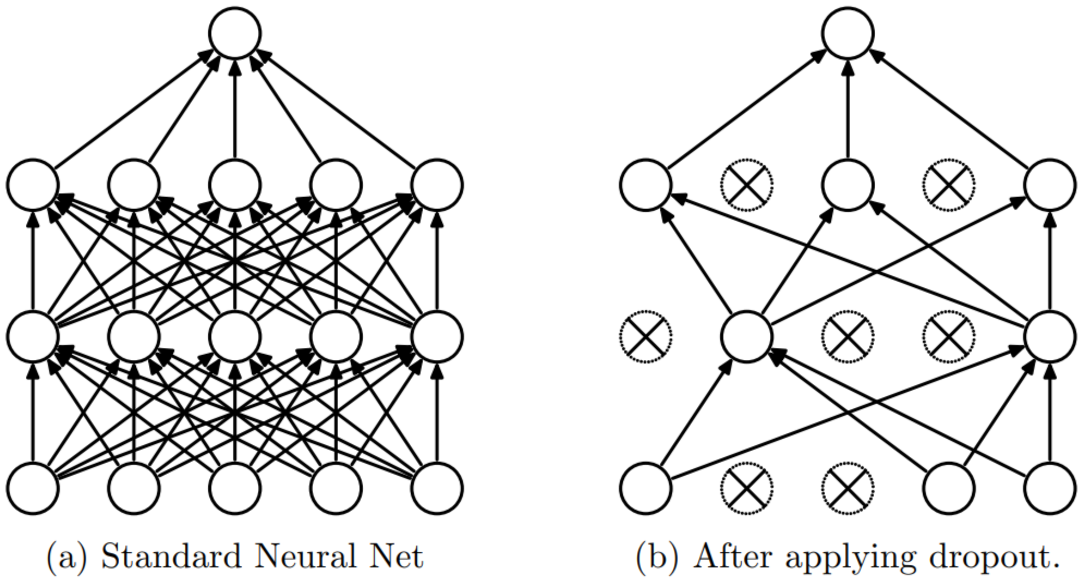


Figure 2.5: Dropout example on a Neural Network.

2.2.2 Transfer Learning

Transfer Learning works by training a network on a big dataset and then using those weights as the initial weights in a new classification task. Typically, just the weights in convolutional layers are copied, rather than the entire network including fully-connected layers. This is very effective since many image datasets share low-level spatial characteristics that are better learned with big data.

2.2.3 Fine-Tuning

Fine-Tuning is conceptually very similar to transfer learning. In Fine-Tuning, the network architecture is defined and then trained on a big dataset. This differs from Transfer Learning because in Transfer Learning, the network architecture must be transferred as well as the weights.

3 Multi-object tracking performance metrics

Content

3.1 Introduction

To proper evaluate the MOT system presented in this work, different performance metrics were used. In the tracking by detection system, for the detection phase the Mean Average Precision (MAP) metric was utilised. With the joining of the detection and the tracking the MOT metrics were used. For the End-to-End network the MAP metric was used to understand the network performance. In the training phase instead of giving an Mean Absolute Error (MAE) to the End-to-end, a Mean Square Error (MSE) was used. The different metrics used to validate the results on both systems will be discussed in the next section.

3.2 Accuracy

Accuracy is normally used in CNN to validate results. Usually if the network has more than one class, the samples of each class must be in equal number. For example if we have a cluster of class A and B and for the class A we have 97% of the samples and for class B we have 3% of sample, if the network has high accuracy on class A it can achieve 97% accuracy but if the samples change to be 60% class A and 40% class B the accuracy drops to 60%.

$$Accuracy = \frac{CorrectPrediction}{TotalInputs} \tag{3.1}$$

3.3 Confusion Matrix

Confusion Matrix, shown in Table 3.1, is a performance measurement used in machine learning classification problem in which output can be two or more classes. It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC Curve.

Table 3.1 Confusion Matrix

		Pred. class	
		P	**N**
Actual class	**P**	TP	FN
	N	FP	TN

where, **P**-positives **N**-Negatives **FN**-False Negatives **FP**-False Positives **TN**-True Positives **TP**-True Positives

With the information of the table we can then calculate:

$$Specificity = \frac{TN}{TN + FP} \tag{3.2}$$

Specificity gives the proportion of actual negatives that are identified as negative.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

Recall gives the proportion of actual positives that are identified as correct.

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

Precision gives the proportion of positive identification that are correct.

3.4 Mean Absolute Error

MAE is the average of the difference between the Original Values (y_j) and the Predicted Values ($\hat{y}_j$). For interpretation reasons its better, however it doesn't consider error directions.

$$MAE = \frac{1}{N}\sum_{j=1}^{N} \mid y_j - \hat{y}_j \mid \tag{3.5}$$

3.5 Mean Square Error

MSE is quite similar to MAE, the only difference being that takes the average of the square of the difference between the original values (y_j) and the predicted values ($\hat{y}_j$). As, we take square of the error, the effect of larger errors become more pronounced then smaller errors, hence the model can now focus more on the larger errors.

$$MSE = \frac{1}{N}\sum_{j=1}^{N} (y_j - \hat{y}_j)^2 \tag{3.6}$$

3.6 Multiple Object Track Precision/Accuracy

Is common to find tracking approaches presented without quantitative evaluation, while many others are evaluated using varying sets of more or less custom measures. With the need of generally applicable metric, the Multiple Object Track metrics proceed to detect the basic types of errors produced by multiple object trackers and introduces two novel metrics, the Multiple Object Tracking Precision (MOTP), and the Multiple Object Tracking Accuracy (MOTA), that intuitively express a tracker's overall strengths and are suitable for use in general performance evaluations.

MOTP is the total error in estimated position for matched object-hypothesis pairs over all frames, averaged by the total number of matches made [19]. It shows the ability of the tracker to estimate precise object positions, independent of its skill at recognizing object configurations, keeping consistent trajectories, and so forth. Where c_t denotes the number of matches in frame t and $d_{i,t}$ is the bounding box overlap distance of target i with its assigned ground truth object. MOTP gives the average overlap between all correctly matched hypotheses and their respective objects and ranges between t_d := 50% and 100%

$$MOTP = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t} \tag{3.7}$$

MOTA is the overall accuracy of how well the system has performed in which m_t, fp_t, and mme_t are the number of misses,of false positives, and of mismatches, respectively, for time t and g_t the number of objects present at time t

$$MOTA = \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \tag{3.8}$$

In MOTA exists three different ratios:

Ratio of misses in the sequence, computed over the total number of objects present in all frames:

$$\overline{m} = \frac{\sum_t m_t}{\sum_t g_t} \tag{3.9}$$

Ratio of false positives:

$$\overline{fp} = \frac{\sum_t fp_t}{\sum_t g_t} \tag{3.10}$$

Ratio of mismatches:

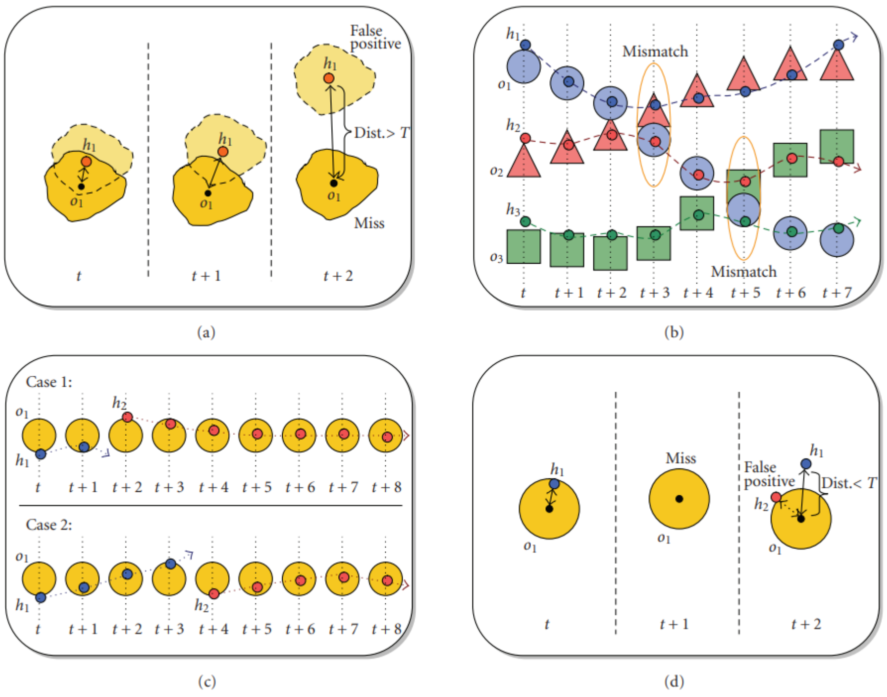


Figure 3.1: Overall of error measures. a) If the object detection exceeds a certain threshold T, σ_1 is considered missed and h_1 becomes a false positive. b) Mismatched tracks. c) For both cases the the mapping would pair h_2 and σ_1 giving different errors, for each case. In case 1 we have 2 errors and in case 2 we have 4 errors, however both cases have only 1 error of the same kind. d) Correct reinitialization of a track. The correspondence is made with h_1 although h_2 is closer to σ_1, based on the knowledge of previous mappings up to time t + 1

$$\overline{mme} = \frac{\sum_t mme_t}{\sum_t g_t} \tag{3.11}$$

To understand the measure performance not by how often mismatches occur, but by how long the tracker correctly identifies targets it was measured the Identification F1 score (IDF1) score. This score are built on top of this truth-to-result match. These scores then measure the number of mismatched or unmatched detection-frames, regardless of where the discrepancies start or end or which cameras are involved.

To address these issues ground-truth identities are first matched to computed ones, more specifically each computed trajectory is associate which exactly one ground-truth trajectory. This score than measure the number of mismatched or unmatched detections-frmaes regardless of where the discrepancies start or end or which cameras are involved.

Let τ(t) be the sequence of detection for true trajectory τ , one detection for each frame t in the set T_τ over which τ extends, and define γ(t) for t $\in T_\gamma$ similarly for computed trajectories. The two simultaneous detection τ(t) and γ(t) are a miss if they do not overlap in space, specifically,

$$m(\tau,\gamma,t,\Delta) = 1 \tag{3.12}$$

When the area of the intersection of two detection box is less than Δ (0 <Δ <1) times the area of the union of the two boxes is declared as a miss. If there is no miss equation (2.4) is equal to zero. If either τ or γ is an irregular node the detection in other trajectories are misses. If both γ and τ are irregular m is undefined.

Was used the Identification False Negatives (IDFN), Identification False Positives (IDFP), Identification True Positives (IDTP) counts to compute Identification Precision (IDP), Identification False Recall (IDR), and the corresponding IDF1. Where F1 score is the ratio of correctly identified detections over the average number of ground-truth and computed detections. More specifically,

$$IDFN = \sum_{\tau\varepsilon AT}\sum_{t\varepsilon T_\tau} m(\tau,\gamma_m(\tau),t,\Delta) \tag{3.13}$$

$$IDFP = \sum_{\gamma\varepsilon AC}\sum_{t\varepsilon T_\gamma} m(\tau_m(\gamma),\gamma,t,\Delta) \tag{3.14}$$

$$IDTP = \sum_{\tau\varepsilon AT} len(\tau) - IDFN = \sum_{\gamma\varepsilon AC} len(\gamma) - IDFP \tag{3.15}$$

Where AT and AC are all true and computed identities in matched ground-truth trajectories and matched computed trajectories, respectively.

$$IDP = \frac{IDTP}{IDTP+IDFP} \tag{3.16}$$

$$IDR = \frac{IDTP}{IDTP+IDFN} \tag{3.17}$$

$$IDF_1 = \frac{2IDTP}{2IDTP+IDFP+IDFN} \tag{3.18}$$

4 State-of-the-art

Content

4.1 Tracking by Detection System

Tracking-by-detection is as the name indicates a tracking system from the detected data. Basically, objects are detected in a frame through features that can be edges, blobs, corners or combination of both. In the detection phase, the system will acquire detected boxes position in the image. After the detection, the tracker will match each object throughout feature identification, overlapping or the weight of booths.

The concepts of speed estimation will be introduced and a brief overview of some of state-of-the-art detectors, trackers and camera calibration approach's studied will be present in this Chapter.

4.1.1 Camera calibration

The camera calibration aims to determine the geometric parameters of the image formation process, a crucial process when metric information is required. New approaches to provide it with less human intervention as possible. These approaches can be divided into two categories, automatically and supervised.

The usual approach to calibrating traffic cameras is normally achieved by acquiring road information like measures of the road and measures of the white lines in the road. This approach is called supervised since it needs some human intervention.

In the automatic approach, the camera can be calibrated trough the movement of vehicles. This approach is normally less accurate but has the advantage of having no human intervention. Each approach will be better explained with examples above.

4.1.1.A Automatic

Dubska [20], presented a method in which they calibrate a camera fully automatically. This method automatically determine 3 orthogonal vanishing points. Then constructs vehicle bounding boxes and automatically determines the camera scale by knowing the statistics of vehicle dimensions.

The 1^{st} Vanishing point direction is parallel to the vehicle motion, marked red in Figure 4.1. It is recovered from detected feature points on the vehicles using Hough transform based on the parallel coordinates and Kanade-Lucas-Tomasi (KLT) tracker. The Hough transform method maps the entire 2D projective plane into a finite space called diamond space 4.1, by a piecewise linear mapping of straight lines in parts.

The 2^{nd} vanishing point direction is perpendicular to the 1^{st} vanishing point direction and parallel to the road. Edges on the vehicles vote in the accumulation space to estimate the direction of the 2^{nd} vanishing point. An edge background model is used in order to select only edges on moving objects. Also the edges with approximately vertical direction are omitted from voting, based on the assumption of scene horizon being approximately horizontal.

The 3^{rd} vanishing point, the camera's intrinsic and extrinsic parameters can be retrieved by assuming the Principal point in the centre of the image.

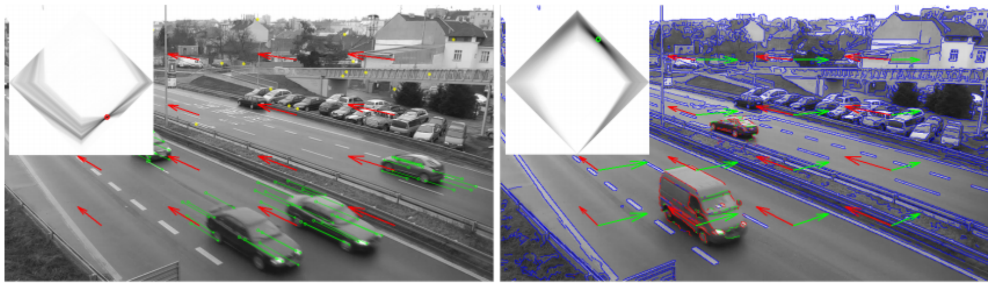


Figure 4.1: (Left) Tracked points using KLT to determine 1^{st} vanishing point. (Right) Calculations for the 2^{nd} vanishing point take strong horizontal edges from vehicles and the vertical edges are discarded.

4.1.1.B Supervised

A method presented in [21] defined vanishing lines in the image through an operator and then calculate the homographic matrix. With this information, the authors can transform image coordinates to the real world.

The most used methods takes measures on the road (white lines, the width of the lane or vehicles measures) to achieve the camera calibration and obtain real-world measures. In studies such as [1], [11], [22] and [23], landmarks are used in the road to calibrate the camera and obtain speed estimations, assuming that the road is on a plane. Others methods already calibrated the camera to collect speed estimation measures and used the time difference between frames, as demonstrated in [24], [25] and [26].

4.1.2 Detection

In this section the detection phase will be discussed, an algorithm that is used to search for the vehicles in video frames, afterwards, that algorithm will identify features of vehicles. Detecting objects can be achieved from multiple ways, either by the older ones, where Harris corner detector was used, identifying interest points in the image, either by Convolution neural networks that consist of several layers with small neuron groups, each of them perceiving small parts of an image. The results from all the neuron groups in a layer partially overlap in a way to create the entire image representation.

4.1.2.A Background subtraction

Some solutions use background subtraction [8] in which the detector acquires blobs in the image through frame differences and uses pre-processing filtering operations to detect objects in the image.

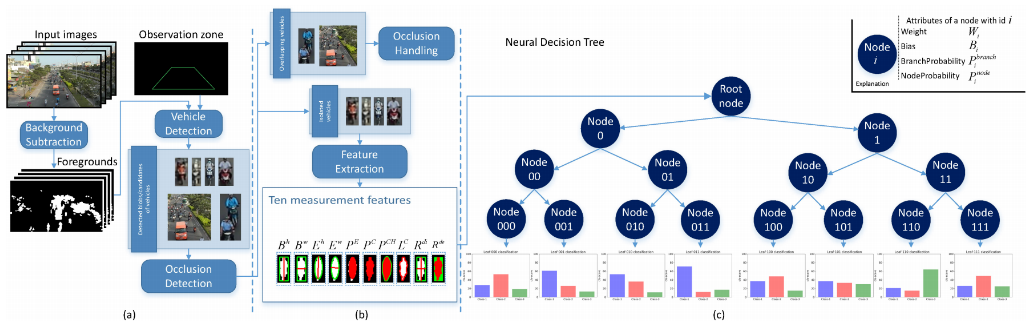


Figure 4.2: Vehicle classification process. (a) Vehicle Detection. (b) Feature Extraction. (c) Vehicle Classification via neural decision tree.

For this method, the authors found some problems with occlusion and camera jitters that were over passed using filters, resulting in low accuracy rates but with a almost real-time image processing. Figure 4.2 shows the method proposed. Concerning background subtraction [27] [24], some authors presented a similar method in which they included sparse representation and low-rank background modelling. In the algorithm the authors had to overcome problems such as noise, illumination changes, shadow moving, waving leaves and branches, resulting in a not so high accurate system and also with delay in real-time applications.

4.1.2.B Single Shot Detector

Other methods Presented in [28], in which a Single Shot Detector (SSD) and an appearance embedding integrated were used.

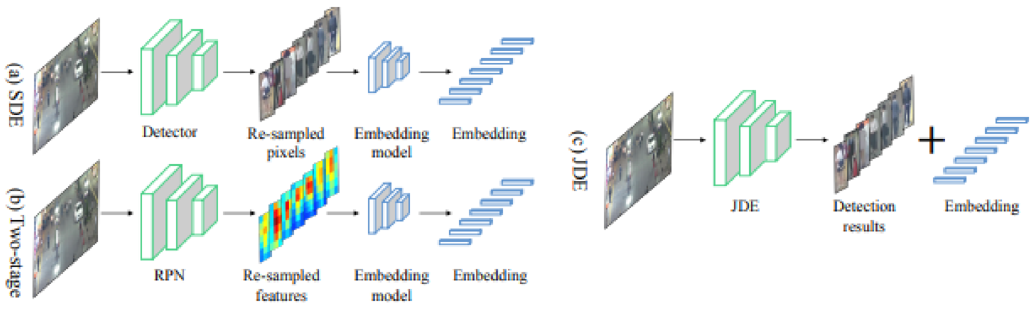


Figure 4.3: Comparison between: (a) SDE model; (b) two-stage model; (c) the proposed Joint Detection and Embedding (JDE).

This method reduced significantly the run time of a MOT system. In that study the authors encountered problems regarding overlapping objects. However, an almost real-time tracking by detection system was achieved (22 frames per second (fps)). On Chapter 5 will be better described, since it was the method used for the detection phase on the tracking by detection network.

4.1.2.C Faster Regional Convolutional Neural Network

CNN methods for detecting objects has presented better performance, as described in [1]. They used three different Faster Regional Convolution Neural Network (RCNN) to identify different parts of the vehicle (front and back; side of vehicle; tiny vehicles far from the camera). Figure 4.4. With that configuration, a detection rate of 88% was obtained. There are others related papers that used CNN on detection phase such as [23], [15], [21], [9], [5] and [25].

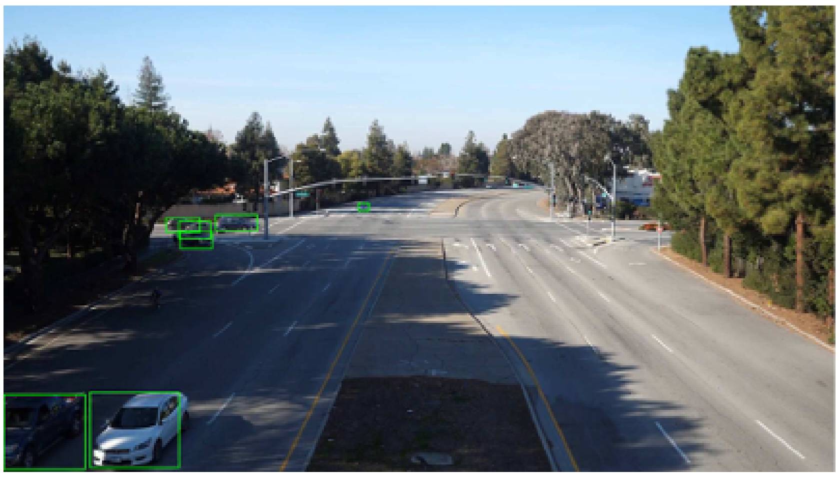


Figure 4.4: Vehicle detection with three different F-RCNN proposed in [1]

4.1.2.D YOLO

Referring to convolution networks, another method of achieving detection of objects is using YOLO, a popular network who achieved very fast detection rates. YOLO has been widely used for vehicle detection [29], [30], [2] and [31]. In [31] the authors implemented YOLO-tiny, using a single convolution layer and a modified version of YOLO. With this implementation, a real-time system was obtained using a budget graphic card (GTX 950M). The study presented in [32] has used YOLOv3-live through YOLOv3-tiny in which the authors passed a Taylor filter through a network layer structure and quantified the network parameters, resulting in a reduction of computational complexity in embedded devices.

4.1.3 Tracking

After the detection phase, some methods use tracking algorithms to follow the identified object in order to obtain the average speed estimation of the vehicles. The methods used for tracking mainly use two different types of tracking, Overlapping and Feature association, which will be discussed above.

4.1.3.A Overlapping

The most used tracking algorithms overlaps a previous detected box with the new detected box (Figure: 4.5). The study presented in [1] tracks vehicles by overlapping the detected boxes, in which the boxes with the larger overlapped area are the right one.

Others studies such as [31], [32], [27] and [29] after detecting objects, they predict on where it supposed to be in the next frame and do an Intersection Over Union (IoU) with the bounding boxes, the boxes with the larger overlapped area is the identified vehicle.

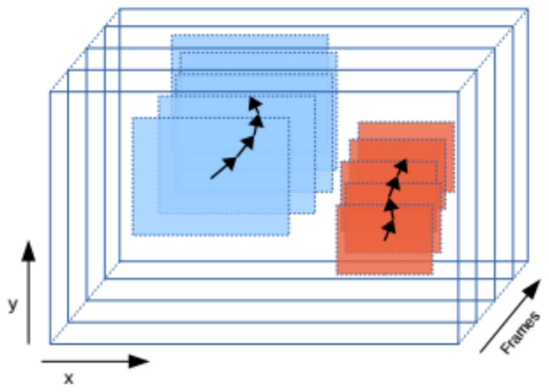


Figure 4.5: Tracking by detection example.

4.1.3.B Feature Association

Instead of tracking boxes, in the studies presented by [24] and [26] it has been tracked the centroid of each object. Another way of tracking objects used by [8], [23], [11], [22], [2], [25] and [30] is to track vehicles using features: after or during the detection phase the authors saved the features of the objects and then they compared them with the new features in the new frames (tracking by detection using different descriptors).

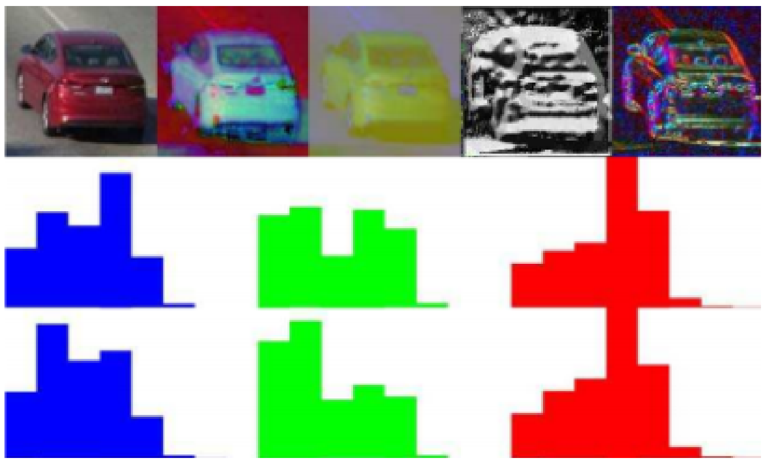


Figure 4.6: Features descriptors utilized by [2]. RGB, HSV, Lab, LBP and gradient feature (first row, from left to right). Second row shows an original color histogram. Third shows the Gaussian spatially weighted histograms where the contribution of background area is suppressed.

Similar to this methods, some studies used deep learning algorithms to obtain the features of the objects and with it better features with lower computational needs were achieved as the results presented in the study [14].

4.1.3.C Feature and Overlapping

The study presented in [7], attached two methods for tracking vehicles that consist of optical flow with IoU of bounding boxes. A study that attached features extracted through deep learning and IoU on bounding boxes with Kalman filter to track vehicles was presented by [21]. The study presented in [33] shown a MOT based on a re-identification task, the authors combined appearance and temporal features. Each object had a tracklet that was generated by appearance similarity with CNN features and IoU.

4.2 End-to-end Network

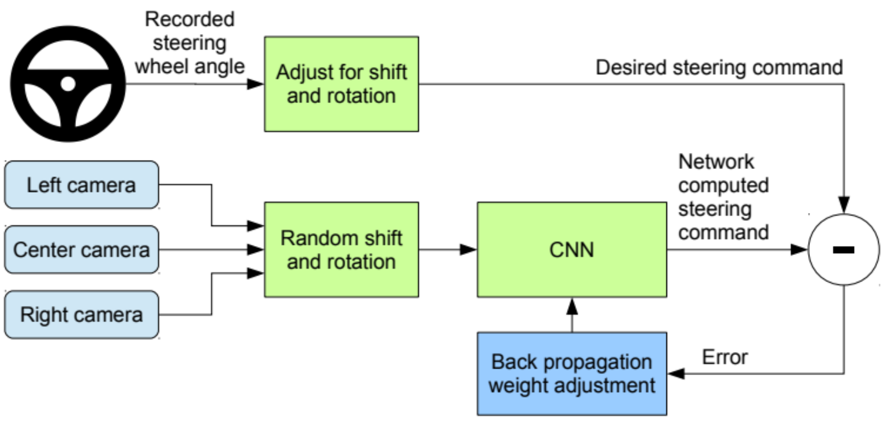

Recorded steering wheel angle
Adjust for shift and rotation
Desired steering command
Left camera
Center camera
Right camera
Random shift and rotation
CNN
Network computed steering command
-
Back propagation weight adjustment
Error

Figure 4.7: Training the neural network.

End-to-end networks have been known in the deep-learning community since the beginning, however it needs much more computational power and data quantity. Currently, with the increase of computational power, deep neural networks can be more like end-to-end networks which can resolve complex problems. For that reason, the state-of-the-art of speed estimation with end-to-end networks is limited, with few works using it. In this dissertation, we will study an implementation that uses an end-to-end Network in order to understand what can be expected and improved with this complex method in Chapter 5.

One of the most known examples is autonomous driving, where all information from cameras, handles, radars, etc. are processed into neural networks to filter out important data and then use it as input to a complex neural network that will predict the next action [34].

The authors of [34] present an end-to-end system for controlling an autonomous car, the system was able to learn useful road features with the use of a convolution neural network in which this method is known to be a very powerful in image recognition task.

The system architecture consists of 9 layers, 5 convolution layers, 3 fully connected layers and 1 normalization layer. With approximately 72 hours of driving data. The network was able to learn how to drive the car on different types of roads and weather conditions.

An end-to-end network has fewer parts in the composition of the system, which can lead to fewer cumulative errors, providing better system accuracy.

5 Methodology

Content

5.1 Velocity Estimation by Object Detection and Tracking

The Velocity Estimation by Object Detection and Tracking method proposed by this work is divided in three different phases:

- The first phase, uses a calibration algorithm to calculate camera extrinsic and intrinsic parameters by detecting the three orthogonal vanishing points from vehicles motion direction.
- The second phase, detects vehicles with a Regional Proposal Network (RPN), this network is previously trained to identify vehicles in each frame and output the detection box of each vehicle with the respective embedding.
- The third phase, outputs from the RPN are fed on the tracker that will identify each object on each frame with the respective ID.

With the identification of each object in each frame, the speed estimation its calculated knowing the mean length dimensions of vehicles. Each phase will be further explained in more detail in the next sections.

5.1.1 Calibration

The camera calibration method used was already implemented in Instituto de Sistemas e Robótica (ISR). This method of Vanishing Point (VP) estimation was based on [20]. It uses a convenient parameterization of lines for the Hough transform, using an accumulation scheme denoted as diamond space, Figure 5.2. This method is fully automatic, which means that there is no need of camera parameters. The scale inference is previously input into the algorithm construction, in this case, the median length measure of a vehicle.

The Calibration started by calculating two VP, the first VP with direction parallel to the vehicle motion, marked as red in Figure 5.1. The incoming video stream was processed frame by frame. On each frame, feature points were collected using Shi and Tomasi's Good Features to Track [35], using the minimum eigenvalue algorithm. Hence, after the corner features were detected, a KLT feature tracker [36] tracked those corners in the subsequent frame.

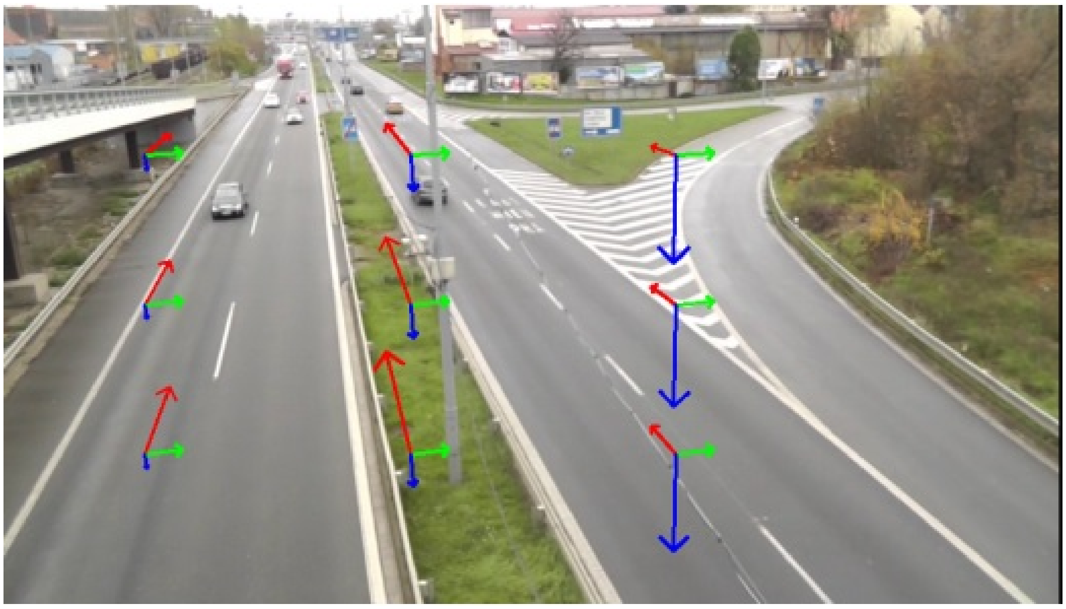


Figure 5.1: Vanishing vectors grid representation

As the observation and accumulation period increases, the first VP is considered very stable and accurate. The first VP it is the most voted point backprojected to the image plane.

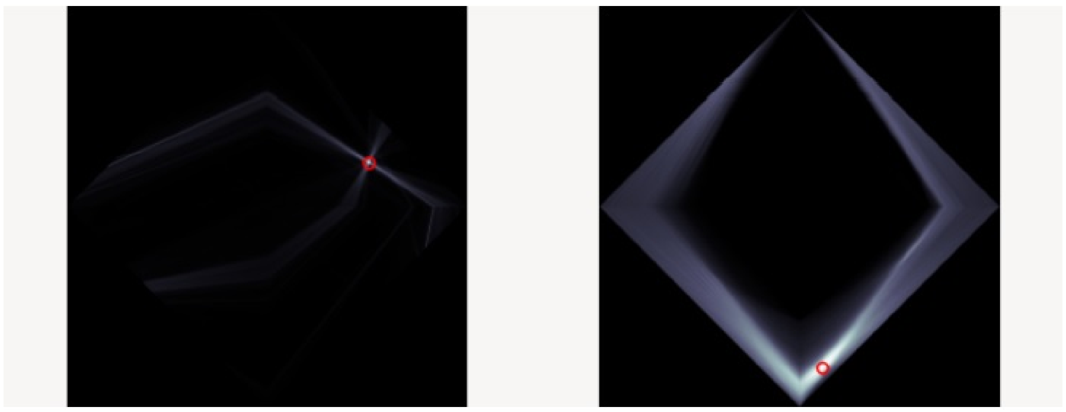


Figure 5.2: Diamond spaces for the Figure 5.1 first (a) and second (b) vanishing points. The red circle signalizes the most voted point in each accumulation space.

For the second VP direction detection, marked as green on Figure 5.1, another diamond space was used. Starting with the assumption that many vehicle edges coincide with the second VP direction. This edge lines were filtered by excluding the ones with high vertical component, then the ones that pass the filtering operation vote in the accumulation space. In order to detect edges on moving vehicles a background edge model was used. In each frame, the model was updated to deal with shadows and other slow lightning changes. For each pixel, the confidence level/degree of an oriented edge occurrence is stored by the edge background model. The points had to meet the following conditions:

- It had to be detected by a Canny edge detector.
- The confidence that the point belongs to the background has to be lower than a predefined threshold, meaning it belongs to moving vehicles.
- The magnitude of the gradient has to be higher than a predefined threshold.
- The line must not be directed towards the first VP.
- The line must not be vertical.

The remaining edges were then extended to infinite lines, those closer to the first VP are excluded from further processing. The line which contains the principal point(assumed in the centre of the image) and is perpendicular to the line defined by the principal point and the first VP has to separate the second VP from the first one.

Assuming the Principal point is in the center of the image, being the first VP denoted as $U = (u_x, u_y)$, the second VP as $V = (v_x, v_y)$ and the principal point as $PP = (p_x, p_y)$. The focal length f can computed accordingly to Equation (5.1). The third point W can be calculated by equation (5.2).

$$f = \sqrt{-(U - PP).(V - PP)} \tag{5.1}$$

$$W = (U - PP) \times (V - PP) \tag{5.2}$$

5.1.2 Regional Proposal Network with Embedding

For the detection, a JDE presented by [28] was used for extracting vehicles detection and features from an image. This method employs a single network to simultaneously output detection results and the corresponding appearance embedding of the detected boxes. Chosen over Separate Detection and Embedding (SDE) because this methods, separate the detection and the embedding in two different networks, increasing its processing time. Figure 5.3.

An input video frame first undergoes a forward pass, through a backbone network to obtain feature maps at three scales. Then, the feature map with the smallest size was up-sampled and fused with the feature map from the second smallest scale and the same was done for the other scales.

Finally, Prediction heads consisting of several stacked convolution layers were added onto the fused feature maps for all scales. The learning objective of each prediction head in JDE can be modeled as a multi-task learning problem.

The JDE method in [28], was implemented for detecting pedestrians. To toggle the problem of detecting vehicles, anchors that fitted vehicles were calculated. The anchors for the vehicle detector were generated through the dataset of UA detrac training. These anchors were calculated through the most common shapes/sizes from the training ground truth using k-means clustering. The number of anchors is set to 12 such that for each scale exists 4 anchors. The network was validated by testing MAP with the change of the IoU, confidence and Non-Maximum Suppression (NMS) values.

The Feature Pyramid Network (FPN) architecture with multiple prediction head estimate predictions from multiple scales, increasing detection accuracy of targets in different distances of the camera, Figure 5.4.

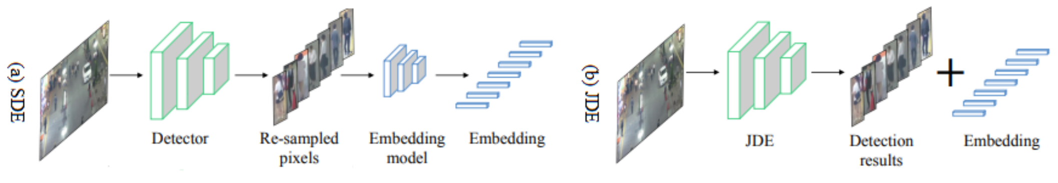


Figure 5.3: Comparison between (a) SDE model, (b) JDE.

JDE method had two objectives: Detect objects accurately and simultaneously output object descriptors. This last objective was achieved by getting a feature that increased distance between objects of different identities and smaller distance between same objects on consecutive frames. This distance can be the Euclidean distance or the cosine distance.

The training of the network was modeled as a multi-task learning problem with anchor classification, box regression and embedding learning. The learning objective of detection had two loss functions, namely the foreground/background classification loss L_α, and the bounding box regression loss L_β. L_α were formulated as a binary cross-entropy loss (5.3) and L_β (5.4) as a smooth-L1 loss .

$$L_\alpha = \frac{-1}{M} \sum [y^i_j log(p^i_j) + (1 - y^i_j) log(1 - p^i_j)] \tag{5.3}$$

in which p^i_j was the predicted probability value of the pixel at location j of example i, y^i_j was a discrete variable indicating the true class of the raw pixel at location j of example i and M was the number of pixels in the example.

$$L_\beta = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise.} \end{cases} \tag{5.4}$$

where t_i is the ground-truth and p_i the probability for the i^{th} class.

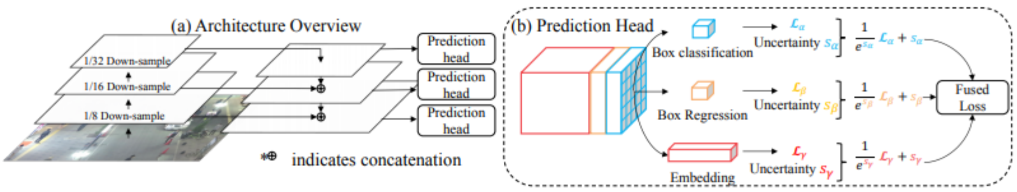


Figure 5.4: (a) The network architecture and (b) the prediction head. Prediction heads were added upon multiple FPN scales. In each prediction head the learning of JDE was modeled as a multi-task learning problem.

The second objective of this network was to learn an embedding feature With attention to the distance between objects i.e., in different frames the same objects were closer between them and other objects were far apart.

The appearance embedding learn was achieved by solving the problem in which the objects with the same identity were closer to each other than the objects of different identity, that were far apart. The training of the network needed as input the identity of each object and bounding box. To achieve this, the (5.5) function was applied.

$$L_{CE}(L_\gamma) = -\log \frac{\exp(f^T f^+)}{\exp(f^T g^+) + \sum_i \exp(f^T g_i^-)}, \tag{5.5}$$

where we denote the class-wise weight of the positive class (to which the anchor instance belongs) as g^+ and weights of negative classes as g^-. f^T represents an instance on a mini-batch, f^- represents a negative sample and f^+ represents a positive sample.

To achieve the multi-task learning problem, a weighted linear sum of losses (5.6) from every scale and every component was applied.

$$L_{total} = \sum_{i}^{M} \sum_{j=\alpha,\beta,\gamma} \frac{1}{2} \left(\frac{1}{e^{s_j^i}} L_j^i + s_j^i \right) \tag{5.6}$$

where, M is the number of prediction heads and s_j^i is the task-dependent uncertainty for each individual loss that balance the importance of each individual task, j=α, β, γ are loss weights for each task.

5.1.3 DeepSORT

The DeepSORT tracker was an improvement of the Simple Online and Realtime Tracking (SORT) tracker, the main difference is the addition of a deep network that can get features from object detection and use them to getting a better performance on tracked objects. SORT performs Kalman filtering in image space and frame-by-frame data association using the Hungarian algorithm with an association metric that measures bounding box overlap.

In our method, to achieve the tracking of vehicles in the image, the state-of-the-art DeepSORT tracker was applied. Instead of using the embedding calculated with DeepSORT network, the embedding used for the DeepSORT was the one generated with the JDE. This embedding was joined with the vehicle detector as explained above.

To limit data accumulation, each track had a limited number without a successful measurement association. This counter was incremented during Kalman filter prediction and reset to 0 when the track was associated with a measurement. The number of frames without a successful measurement were counted, and if the number exceed a maximum age A_{max}, the tracks were deleted from the track set. To start a new track, each detection that was not included in the existing tracks was taken as a candidate. If more than three detections were successfully associated in consecutive frames, a new track would start.

To solve the association between the predicted Kalman states and newly successful measurements, an Hungarian algorithm was used. To incorporate motion, the (squared) Mahalanobis distance given by Equation (5.7), between Kalman states predictions and newly arrived measurements were used. The Mahalanobis distance took the state estimation uncertainty into account by measuring how many standard deviations the detection was away from the mean track location [37].

$$d^{(1)}(i,j) = (d_j - y_i)^T S_i^{-1} (d_j - y_i) \tag{5.7}$$

where, the projection of the i-th track distribution into measurement space is given by (y_i, S_i) and the j-th bounding box detection by d_j.

For tracking through occlusions, a second metric, that measure the smallest cosine distance given by Equation (5.8), was integrated into the assignment problem. This way, when the object reappears the system will be able to identify it using the association with appearance descriptor provided by the detector.

$$d^{(2)}(i,j) = min\left\{1 - r_j^T r_k^{(i)} \mid r_k^{(i)} \in R_i\right\} \tag{5.8}$$

where, r_j is an appearance descriptor that is previously saved in R_k between i-th track and j-th detection in appearance space.

An binary variable is introduced to indicate if an association is admissible according to the Equation (5.9)

$$b_{i,j}^{(2)} = 1 \left[d^{(2)}(i,j) \leq t^{(2)}\right] \tag{5.9}$$

In summary, Mahalanobis distance provided information, based on motion, about possible object locations that were particularly useful for short-term predictions. This Mahalanobis distance has a threshold of 95% computed from the inverse χ^2 given by Equation (5.10). The cosine distance considered appearance information that were particularly useful to recover identities after long term occlusions [37]. These two metrics (Mahalanobis distance and cosine distance) were associated with a weighted sum.

$$b_{i,j}^{(1)} = 1 \left[d^{(1)}(i,j) \leq t^{(1)}\right] \tag{5.10}$$

The solving of this problem was applied through a cascade matching method. As input, a maximum age (A_{max}), existing tracks indices (T) and detection indices (D) were given, then the association cost matrix (C), given by Equation (5.11) and the matrix of admissible associations (B) were computed, given by Equation (5.12). Next, over track n was iterated by increasing age to solve the assignment problem.

After, we associated the tracks that had not been associated with a detection in the last frame. Then, the linear assignment problem between tracks that had not been associated and unmatched detection was solved. After, the set of matches and unmatched detection were updated. This sequence method gave priority to tracks that had been seen more recently. To sum up, the unconfirmed and unmatched tracks in the second last frame were associated with IoU and this helped to account for sudden appearance changes. The matching can be seen in Table 5.1.

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda)d^{(2)}(i,j) \tag{5.11}$$

where, λ defines the combined association cost.

$$b_{i,j} = \prod_{m=1}^{2} b_{i,j}^{(m)} \tag{5.12}$$

Table 5.1 Matching Cascade

```
Input:  Track indices $T$ = {1,...,N}, Detection indices $D$ = {1,...,M}, Maximum age $A_{max}$
1:      Compute cost matrix $C$
2:      Compute gate matrix $B$
3:      Initialize set of matches $M \leftarrow \emptyset$
4:      Initialize set of unmatched detection $U \leftarrow D$
5:      for $n \in 1,...,A_{max}$ do
6:          Select tracks by age $T_n \leftarrow i \in T | \alpha_i = n$
7:          $[x_{i,j}] \leftarrow min_cost_matching(C, T_n, U)$
8:          $M \leftarrow M \cup (i,j) | b_{i,j} \cdot x_{i,j} > 0$
9:          $U \leftarrow Unj | \sum_i b_{i,j} \cdot x_{i,j} > 0$
10:     end
11:     return $M, U$
```

5.1.4 Speed Estimation

To estimate speeds from cars we pick the lower middle point from the bounding box and calculate the distance passed of that point on consecutive frames. With focal length and VP's described in the camera calibration Subsection: 5.1.1, we can project points in the image to the ground plane (ϕ).

Being W' the 3rd VP with world coordinates and the camera coordinates $O(p_x, p_y, 0)$, the ground plane vector (n_ϕ) can be obtained with Equation: 5.13. The last parameter (d) for (ϕ) is an arbitrary value. The projection of point $A'(x,y,f)$

in ground plane can be obtained with Equation: 5.14. The relative distance (d_r) between points A' and B' is obtained with Equation: 5.15

$$n_{\phi} = W' - PP \tag{5.13}$$

$$\phi(A') = \phi \cap \overleftrightarrow{OA'} \tag{5.14}$$

$$d_r = |\phi(B') - \phi(A')| \tag{5.15}$$

To estimate speed, the scale factor must be calculated so that the actual metric dimensions can be defined. The method for transforming dimensions on the image to the world dimensions was achieved by fitting statistics of known dimensions and the measured data from the traffic. These statistics information is given in [38]. Instead of using the minimum of the three scales (height, length, width) as presented in [38], the box length of vehicles acquired from the camera calibration method was used.

The scale was then calculated with equation (5.16) being the time pass for each point, obtained from the camera fps. The speed (v) for a point X_i in the ground plane, is obtained with Equation: 5.17 being the value 3.6 the transformation from m/s to km/h.

$$\lambda = \frac{l_c}{l} \tag{5.16}$$

where λ is the scale, l_c is the statistics estimated dimension length and l is the relative length of vehicles in the camera.

$$v = \frac{d_r(X_i, X_{i-1})\lambda}{\frac{1}{\text{fps}}} * 3.6 \tag{5.17}$$

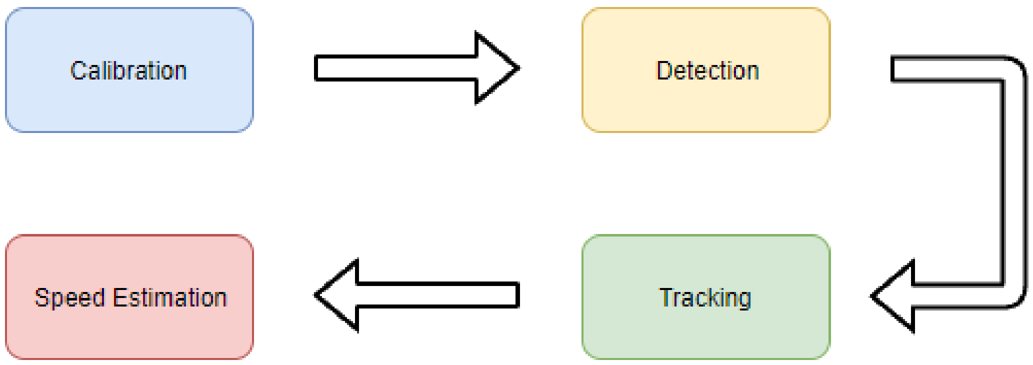


Figure 5.5: Speed estimation using tracking by detection method

5.2 Velocity Estimation Through an End-to-End Network

The End-to-End Network proposed in this work is based on 3-dimensional convolutions networks, that is a modified 3DResnet50 model. This network estimate the average vehicle speed from video samples. The method had 4 different features. Firstly, instead of using a 3D convolution that would be hard to train and time consuming, it used an inflated(2+1)D convolution.

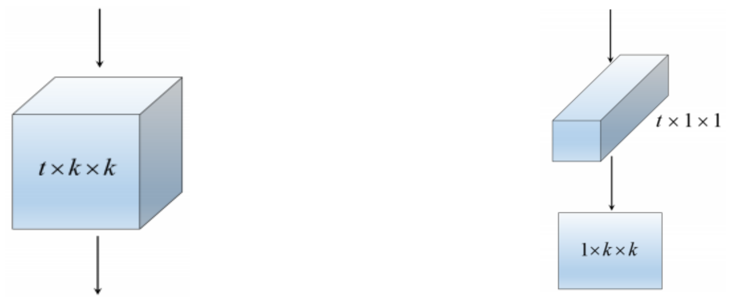
Secondly, non-local blocks were utilized for better capture spatial capture and temporal dependency.

Thirdly, a multi-scale convolutions network was constructed in order to extract information on the various scales differences of vehicles in the image.

Finally, an optical flow was added to the video, thus giving extra information about speed and direction of pixel motion in the image. In the next sections, each part of the network will be explained.

5.2.1 Inflated (2+1)D Convolution

The inflated (2+1)D Convolution consists of 1D temporal operation and a 2D spatial convolution. It has less parameters in comparison to 3D convolution, thus its is easier and faster to train. The 3D convolution kernel was initialized as a 2D convolution kernel, which was pre-trained on ImageNet.



(a) 3D Convolution (b) (2+1)D Convolution

Figure 5.6: Difference between a 3D Convolution and (2+1)D Convolution

In 3D convolution, the time dimension cannot be reduced neither too fast or too slow, because the time dimension relies on image dimension and frame rates. If it is not reduced carefully there is the possibility that the network would confuse the edges of different objects, or it might not capture effectively the scene dynamics.

5.2.2 Non-Local Blocks

The non-local blocks expand the receptive field of the model to capture long-range dependency and more vehicle spatial-temporal features from the video. A non-local operation defined in equation (5.18) computes the response in one position as a weighted sum of the features in all positions. The operation on non-local blocks can be defined as:

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j) g(x_j) \tag{5.18}$$

in which i is the index of an output position whose response is to be computed, j is the index that enumerates all possible positions, x is the input signal, which could be in the form of images, sequences, and video features, while y_i is the output signal of position i. A pairwise function named as f computes the correlation between x_i and all x_j. The function g a representation of the input signal at the position j. The summation of f and g is normalized by a factor $C(x)$. The output behaviour considers all positions ($\forall j$) in the operation.

The pairwise function f 5.19 can be the dot product, the Gaussian, or the Embedded Gaussian. Since the non-locals models are not sensitive to choice of f according to experiments, it has been randomly chosen the Embedded Gaussian. [39]

$$f(x_i, x_j) = e^{\theta(x_i)^T \phi(x_j)} \tag{5.19}$$

in which x is the input signal, i is the index of an output position, j is the index that enumerates all possible positions, $\theta(x_i)$ and $\phi(x_j)$ are two embedding and the normalization factor (5.20).

$$C(x) = \sum_{\forall j} f(x_i, x_j) \tag{5.20}$$

5.2.3 Multi-Scale Convolution

Multi-Scale Convolution was used to overcome the problem where a vehicle closer to the camera appears larger than it really or the other way around. To solve this issue, kernels of different sizes where added to the first layers separately from the network. The outputs are concatenated to feed the input of the next layer, afterwards.

The information with different scales was considered in multiple convolution kernels, making the network more adaptive. This different convolution kernels are shown on Figure 5.7. The output of the different kernels are then concatenated to feed the next as input to the next layer.

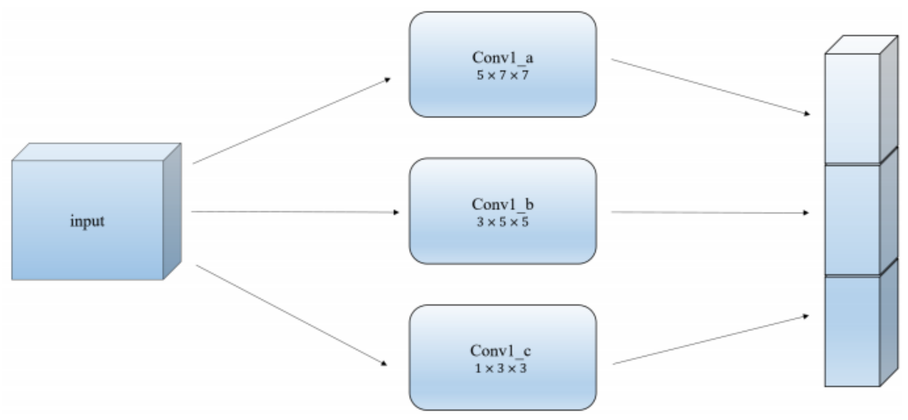


Figure 5.7: Multi-Scale CNN

5.2.4 Optical Flow

The addiction of Optical Flow to the network gives the network the motion patterns of the objects, surfaces and edges. Optical flow can be defined as apparent velocity of brightness patterns in image. Since it reflects the velocity of pixel points, there is a relation between the vehicle's velocity and for that reason optical flow was used, concatenating it with the image as the input. Experiments made by [16] showed a decrease in velocity estimation error by adding this feature.

In this work, the optical flow used in [40], extracts perpixel features from both input images, along with a context encoder that extracts features from only one of the images. A correlation layer that builds multi-scale 4D correlation volumes for all pairs of pixels by taking the inner product of all pairs of feature vectors. The last 2-dimensions of the 4D volume are pooled at multiple scales to construct a set of multi-scale volumes. Finally, an update operator which recurrently updates a flow field through a recurrent unit that performs lookups on the correlation volumes.

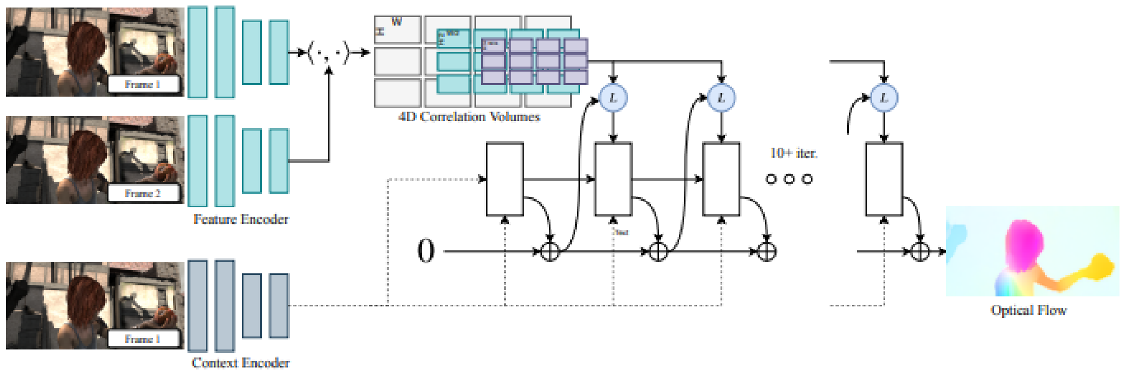


Figure 5.8: RAFT main components: feature encoder, correlation layer and an update operator.

The Network process starts with a pair of consecutive RGB images, I_1, I_2. Then, a dense displacement field was estimated, which maps each pixel in I_2 to its corresponding coordinates in I_1. The method can be distilled down to three stages: feature extraction, computing visual similarity, and iterative updates. All stages are differentiable and composed into an end-to-end architecture.

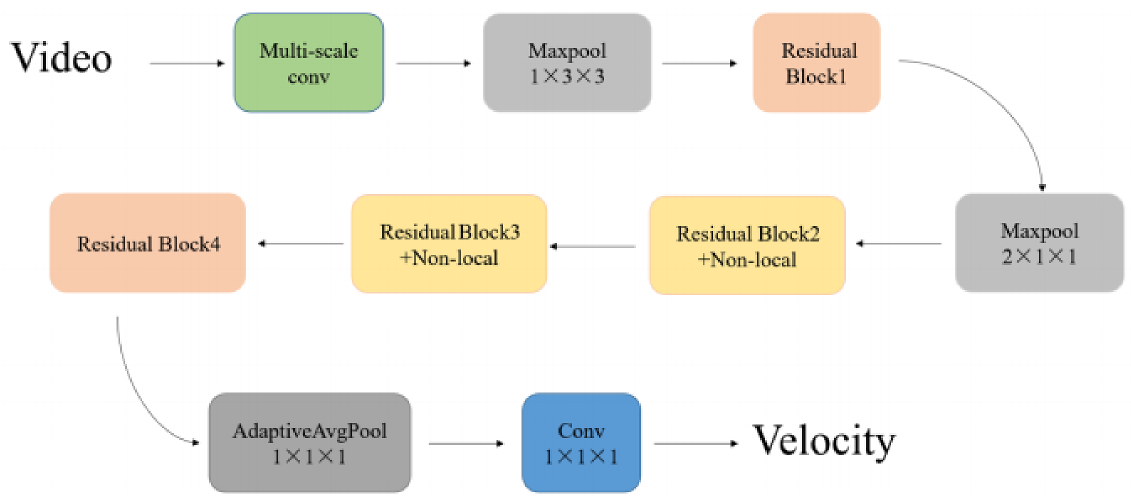


Figure 5.9: End-to-end model architecture

6

Experimental validation and results analysis

Content

Experimental validation and analysis of results are the final proof of each system. Systems need validation before being applied in real-world situations. Validation methods must go beyond the intended purpose of the system to address unanticipated problems and noise during the project development phase. Therefore, the results must be carefully analyzed and discussed for future improvement.

6.1 Datasets setup

In order to train the CNN, the dataset needed to have a ground truth that is related to the specifications of the network that is going to be trained. Since we had two different systems(Tracking by Detection and End-to-end network), the datasets used for each system were different.

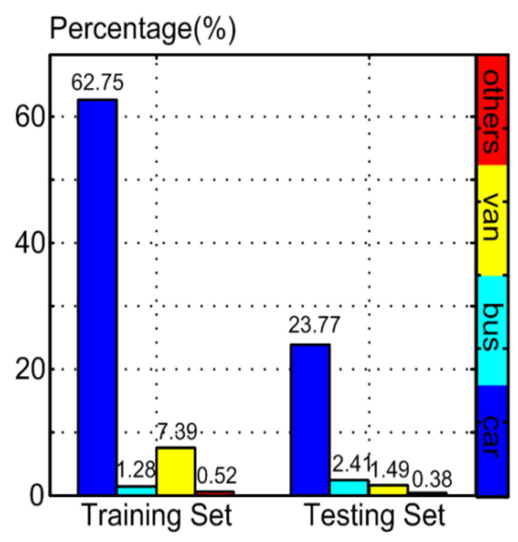
The dataset for the Velocity Estimation with Tracking by Detection had a bounding box ground truth, where each bounding box is characterized with nine values (Frame, Number, Left, Right, height, width, score, type, colour). For the end-to-end Network, the dataset used the median speed between a defined distance.

6.1.1 UA Detrac

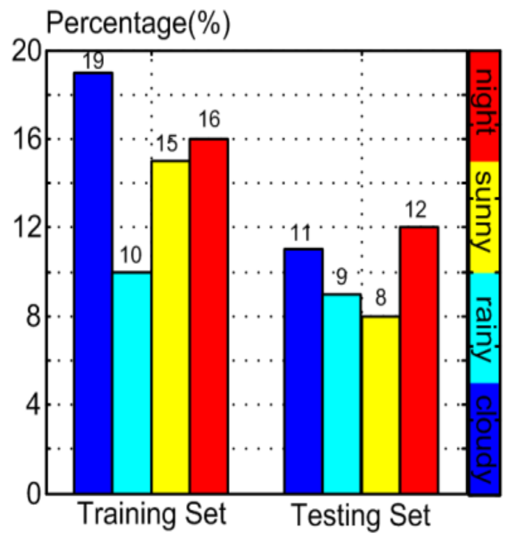
UA Detrac dataset is normally used to train networks for detecting and tracking objects in an image for MOTA results.

The dataset consists of 10 hours of videos captured with a Canon EOS 550D camera at 24 different locations in Beijing and Tianjin, China. The videos were recorded at 25 fps, with a resolution of 960×540 pixels. There were more than 140 thousand frames and 8250 vehicles that were manually annotated, leading to a total of 1.21 million labelled bounding boxes of objects.

It has also been performed benchmark tests of state-of-the-art methods in object detection and multi-object tracking together with evaluation metrics.



(a) Vehicle Type

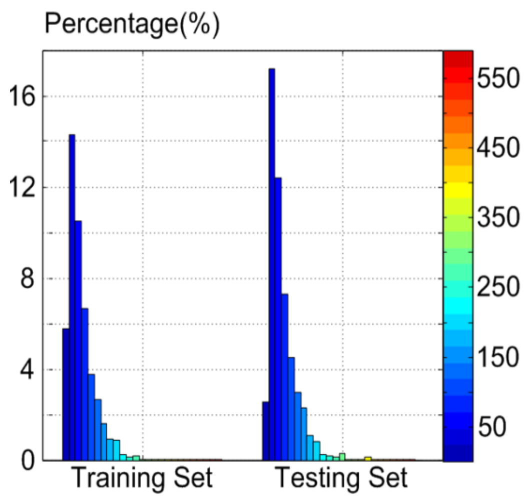


(b) Illumination

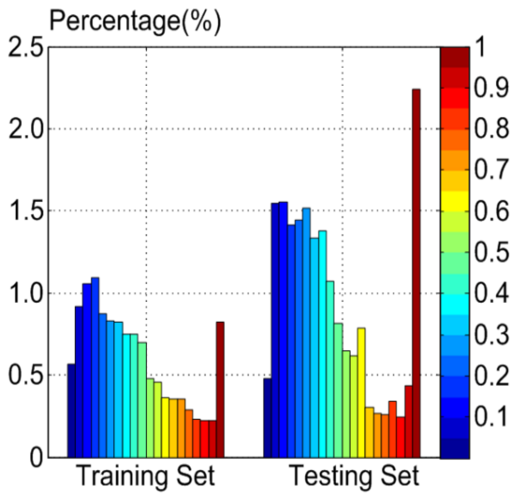
This dataset was labeled with several categories:

Vehicle category , classified into four categories 6.1a: car, bus, van, and others.

Weather with four categories 6.1b: cloudy, night, sunny, and rainy.



(a) Scale



(b) Occlusion Rate

Scale, defined the scales of the annotated vehicles as the square root of their area in pixels. It has three groups of scales 6.2a: small (0-50 pixels), medium (50-150 pixels), and large (more than 150 pixels). Occlusion ratio, the fraction of the vehicle bounding box being occluded that defines the degree of occlusion. The degree of occlusion had three categories 6.2b: no occlusion, partial occlusion, and heavy occlusion. Where partial occlusion is the occlusion ratio of a vehicle between

1%-50%, and the heavy occlusion is an occlusion ratio larger than 50%. Truncation ratio: the truncation ratio refers to the degree of the vehicle parts outside of the frame.

6.1.2 BrnoCompSpeed

BrnoCompSpeed dataset is a high-quality dataset that has the velocity and time of cars between 2 points. The dataset consisted of 18 full-HD (1920x1080) videos (6 sessions on different locations, 3 videos from different angles for each location) and there was a total of 20865 vehicles with known ground truth speed.

The dataset provided high-quality videos with various traffic conditions (low traffic in Session 3, high traffic in Sessions 5 and 6). However, it was quite limited in lighting and weather conditions. Almost all videos were taken in cloudy weather (except for some parts of Session 3) with no distracting phenomenons (fog, rain, etc.).

6.2 Assessing Baseline Performance

In this section, it will be presented how both CNN's were tested before training on a dataset with more data, consequently with more training time.

6.2.1 BrnoCompSpeed

6.2.1.A Velocity Estimation by Object Detection and Tracking

Preliminary tests of the tracking by detection network were performed by using the BrnoCompSpeed dataset. First, we transformed the data in a way that the CNN had all the data organized to be trained. The video was separated in frames in order to accelerate the training. Each frame was associated with a text document that contains the information of the box and index of each car in that frame. As the index and the boxes of the car were on different files, it was necessary to join both.

The tests were performed on a single video, and the results were satisfactory. The detection times were noteworthy (12 fps) and the network could detect and track the majority of the vehicles.

6.2.1.B End-to-end Network

The end-to-end needs a great quantity of data in order to have a good performance. Networks like this one had a greater training time than others, and for that reason, the main work of this network was to understand its behaviour when applied in different views and locations.

The tests performed on the network involved different optical flows in order to analyse any differences between them. The optical tests were made with the RAFT optical flow and Farneback optical flow. Since both had similar results we chose the RAFT optical flow, being this with slightly better MSE (71.59 vs 85.11).

After choosing an optical flow algorithm, the network training and validation started. Firstly, it was trained for one view and the results were validated. The results for mean absolute error were satisfactory(4.53 km/h) although there were some peak values and for that reason the mean square error were not satisfactory(73.59). Nevertheless, the network training continued and it will be discussed on the next chapter.

6.2.2 UA Detrac

The UA Detrac dataset was used to confirm the tests made on BrnoCompSpeed. On the UA Detrac, the ground truth was more complete and consequently more accurate to test CNN for vehicle detection which output boxes on the identified objects.

The tests applied were not successful like those of the BrnoCompSpeed dataset. In the UA Detrac tests, the MAP was lower. An observed fact was that the test dataset from the vehicle side view was not trained. With this knowledge, the test dataset videos that had different orientations in relation to the training and that were not interesting to our work were consequently removed. Those videos were the ones in which the orientation was a 90-degree angle from the cars and in which existed road crossing.

6.3 Assessing validation and results

Different validation methods were used for each type of system component, so we could access the validation results of the overall system. For each type of validation, we need to acquire data from tests and analyse it to understand if some problems or errors could be suppressed, for this it will be presented the way that data was used to check for errors or problems.

Table 6.1 Anchors calculated with UA Dataset used for the RPN 576×320.

Anchors	Width(px)	Height(px)
1	17	13
2	22	18
3	30	22
4	43	23
5	38	32
6	59	29
7	51	45
8	79	37
9	67	61
10	104	52
11	94	84
12	152	80

6.3.1 Velocity Estimation by Object Detection and Tracking

The Velocity Estimation by Object Detection and Tracking can be divided in three parts. For that reason each part of the network was validated differently.

The detection part that used the machine learning network was mainly validated with the training of the network and by observing its performance.

The multiple object tracking depends on the detection accuracy and for that reason the validation included both parts of the network.

The speed estimation of vehicles included the last two parts, giving the overall performance of the network.

6.3.1.A Detection

The Detection phase was validated using the UA Detrac dataset by applying the detection MAP, giving how successful the network was trained by calculating the true positives prediction over the total predicted detection. The anchors calculated for the RPN are shown on 6.1.

The results on 6.2 show the results obtained by training the network on the UA Detrac dataset. This network traine has in account the MAP and the Embedding.

The results that had a different image resolution showed that higher resolution, in our case, did not improve object detection. In this case, it should have improved even for a small change, which led us to think that the anchors on the smallest resolution were giving a more general detection of the object in comparison to the higher resolution ones.

Table 6.2 Results of training with different image resolutions in which the resolution of 576×320 could achieve 12 frames per second and the resolution of 1088×608 could achieve 10 frames per second.

Epoch	576x320		1088x608	
	MAP	Emb	MAP	Emb
0	0.6624	0.2446	0.2071	0.2383
5	0.7372	0.3156	0.6597	0.2832
10	0.6953	0.3419	0.6913	0.3144
15	0.6840	0.3687	0.6782	0.3518
20	0.7437	0.3591	0.7120	0.3997
25	0.6566	0.3441	0.6998	0.4378
30	0.6765	0.3670	0.6778	0.5434
35	0.6662	0.3779	0.7558	0.5639
40	0.6851	0.3810	0.6710	0.4430
45	0.7068	0.3512	0.6588	0.4925

The smallest resolution was better in object detection but was also faster in processing each image as expected in the neural network of object classification because of the lower complexity. For that reason, the resolution chosen was 576x320, where the best results were observed.

6.3.1.B Tracking

For the tracking phase, the dataset used was the UA Detrac dataset. On the results was applied the MOTA metrics used in the UA Detrac repository in order to quantify its performance. The test results weather conditions have 4 different types, the results of each type for the sequences tested in UA-Detrac are shown in Tables 6.3(Cloudy) 6.4(Rainy) 6.5(Night) 6.6(Sunny).

Table 6.3 Results obtained with detection and tracking on Cloudy conditions sequences. It was composed with 520 vehicles in 11 255 frames.

MOTA (%)	IDF1(%)	IDP(%)	IDR(%)	Rcll(%)	Prcn(%)	MOTP(%)
55.8	80.7	69.4	96.4	97.4	70.1	87
15.8	62.9	55.4	72.8	73.7	56.0	80.6
36.2	64.6	55.9	76.4	86.7	63.5	81.6
43.6	67.8	58.6	80.3	90.5	66.0	78.9
54.9	77.5	69.3	88.0	91.1	71.7	78.2
62.3	73.0	66.4	80.9	92.3	75.7	80.2
12.1	60.7	48.4	81.3	90.2	53.7	73.6
81.4	87.5	89.1	86.0	89.0	92.2	79.6
44.3	61.4	55.2	69.2	84.9	67.8	75.5

Table 6.4 Results obtained with detection and tracking on Rainy conditions sequences. It was composed with 728 vehicles in 11 630 frames.

MOTA(%)	IDF1(%)	IDP (%)	IDR(%)	Rcll(%)	Prcn(%)	MOTP(%)
73.3	81.6	81.4	81.8	87	86.6	79.1
21.6	68.3	56.0	87.3	88.9	57.0	68.8
49.3	66.0	69.0	63.3	70.6	77.0	73
41.3	55.5	74.0	44.4	50.8	84.7	82.9
30.7	50.8	70.6	39.7	43.6	77.5	80.2
52.4	67.5	71.5	63.8	71.0	79.5	76.9
54.4	56.2	64.9	49.5	65.7	86.1	73
46.0	56.1	72.0	46.0	55.1	86.4	77.1

Table 6.5 Results obtained with detection and tracking on Night conditions sequences. It was composed with 506 vehicles in 18 469 frames.

MOTA(%)	IDF1(%)	IDP(%)	IDR(%)	Rcll(%)	Prcn(%)	MOTP(%)
46.4	72.5	63	85.4	91.1	67.1	82.9
58.8	79.1	73.3	85.9	88.1	75.1	81.6
71.3	75.7	73.8	77.7	88.4	83.9	80.3
46.7	62.7	74.5	54.1	59.8	82.3	82.9
52.6	75.9	70.6	82	84.5	72.8	79.3
57.5	58.1	61.3	55.3	74.2	82.4	81.3
64.0	77.7	71.0	85.8	92.5	76.5	79.5
69.7	77.5	73.1	82.4	91.3	81.0	85.1
54.6	61.2	74.9	51.7	62.2	90.0	68.6
50.4	54.9	74.1	43.7	54.8	93.0	69.1
36.6	36.7	59.5	26.5	41.0	91.8	67

Table 6.6 Results obtained with detection and tracking on Sunny conditions sequences. It was composed with 237 vehicles in 7 055 frames.

MOTA(%)	IDF1 (%)	IDP(%)	IDR(%)	Rcll(%)	Prcn(%)	MOTP(%)
68.0	83.1	80.2	86.1	88.0	82.1	76,6
44.3	68.7	66.7	70.7	75.5	71.3	78.5
43.4	67.1	60.2	75.7	84.9	67.4	75.7
67.2	75.8	78.0	73.6	80.9	85.7	80.1
48.8	60.1	66.5	54.9	65.9	79.9	79.4

Table 6.7 Overall results obtained with detection and tracking.

	MOTA(%)	IDF1(%)	IDP(%)	IDR(%)	Rcll(%)	Prcn(%)	MOTP(%)
Cloudy	44.3	67.8	58.6	80.9	90.2	67.8	79.6
Rainy	47.7	61.1	71.1	56.4	68.2	82.1	77.0
Night	54.6	72.5	73.1	77.7	84.5	82.3	80.3
Sunny	48.8	68.7	66.7	73.6	80.9	79.9	78.5
Total	48,3	68,3	68,9	75,7	82,7	81	79,1

The results of the tracking by detection on the UA-Detrac dataset were in line with the best methods tested in the UA-Detrac site. The main problem occurs with miss detection were it was observed, that in some frames the network failed to detect vehicles in the image. This problem makes the tracker lose car tracklets, compromising the accuracy of the tracker. To track it with a higher accuracy a better detector should have been used, but doing so would reduce the frame per second performance of the network(10 fps).

6.3.1.C Speed estimation

The speed estimation has been validated with the BrnoCompspeed dataset. The Brnocompspeed dataset had the data of vehicles organized as follows: each car had a time in which it passed the first laser sensor, a time in which it passed in front of the second laser sensor, a flag that was true or false if the speed measures were correct or not, and identified lane of where the car passed and the speed measured.

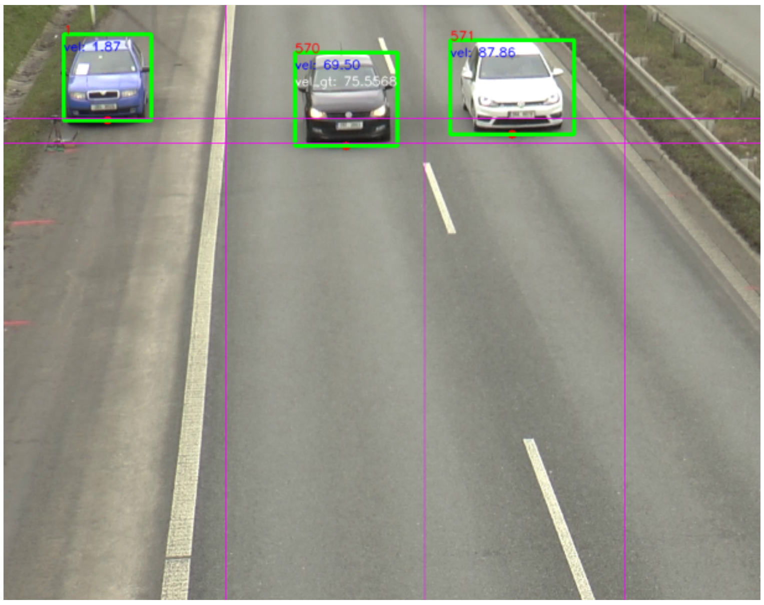


Figure 6.3: Ground-truth join with detection.

In the Figure 6.3, three cars can be seen and two of them pass in front of a motion sensor mounted on the road. The vertical purple lines divide the road into two lanes. The zone limited by the horizontal lines identifies the reading zone of the motion sensor. Each car has an intersection time, given by the motion sensor and a specific velocity. The intersection time was converted to a specific frame.

If a detected box exists in the zone limited by the horizontal lines and in the frame exists a velocity, both will match. This way we can compare the ground-truth speed with the speed estimation.

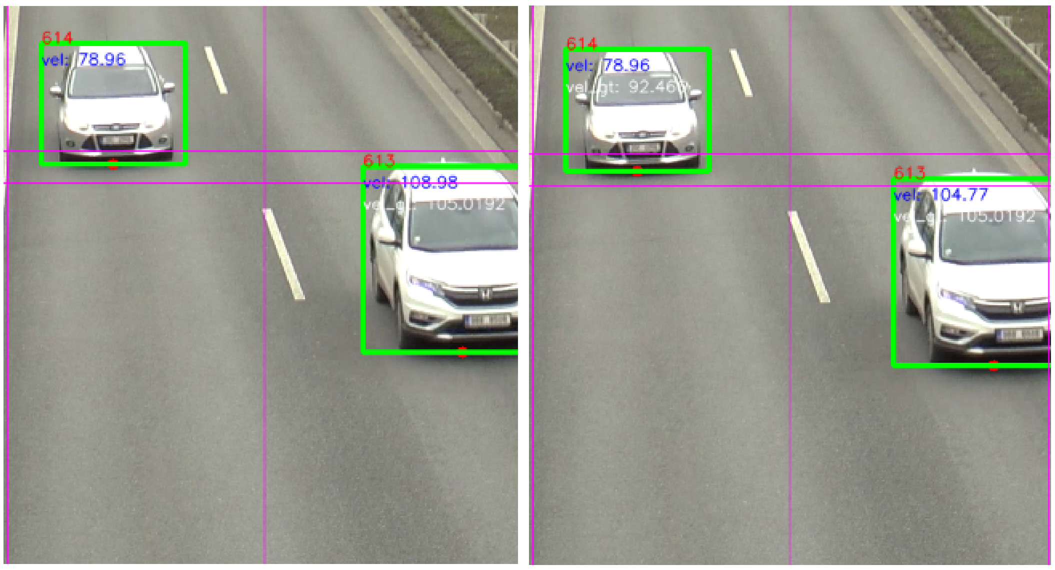


Figure 6.4: Identification of each vehicle from the tracker with the ground-truth

The Figure 6.4 shows the moment in which the vehicle detected (614) was identified. The absolute mean speed estimation error is the difference between the estimation and the ground truth.

The difference between the speed estimation and the ground truth, showed in the image, can be explained by the fact that as the velocity of the ground truth was the median velocity between the two sensors, in this case, it is the instant velocity that was calculated. That is one of the reasons for the difference observed between ground-truth and estimation. Another error that can occur is the increase in the speed estimation error as the objects are further away from the camera. This is related to the camera pixel resolution, the higher the pixel resolution the smaller is the error.

The Results of the speed estimation were obtained by estimating the velocity of the cars in which the detector had a detection between the start and the end of the sensors, which is also used to calculate the speed error. In session 1 and centre view the mean error was 3.64km/h and the max error was 11km/h. This means that for a median velocity of 90km/h in the ground truth the mean error was about 4% and a max error of 12%.

6.3.2 End-to-End Network

For the test of this CNN, the speed estimation accuracy was the main concern. The dataset used to test the CNN was BrnoCompSped with a single video without flow and with 16 frames in 10 sec proposed by 5.9. On this first test, the CNN was not so accurate, consequently. What was observed was that the network was almost giving a general mean speed. To specify the mean speed for fewer cars the network was tested to estimate in two seconds video sequences.

The applied method results in higher accuracy than the previous one. Following the better results with lower time for each video, the training of the network was made with that improvement. This improvement suits better for our case because it gives a speed estimation closer to the mean speed for each car.

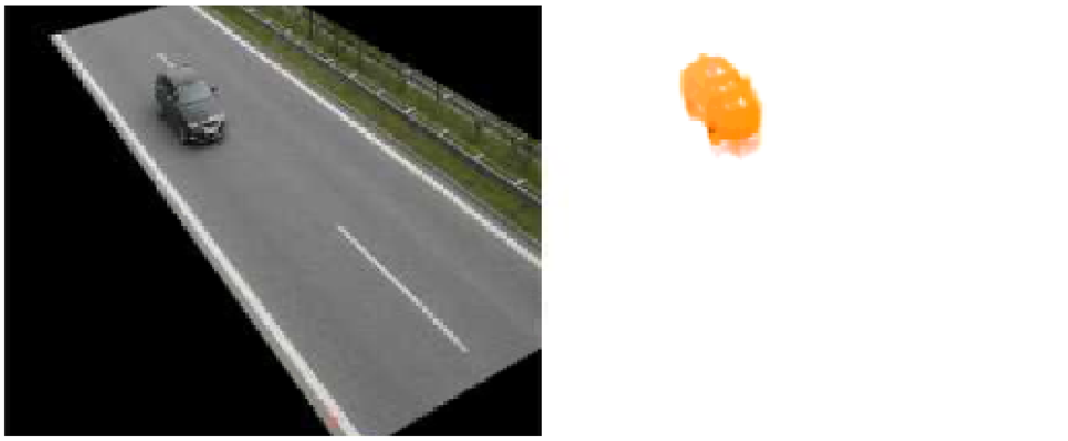


Figure 6.5: RGB Image (3D) with RAFT Optical Flow (2D).

The tests were mainly applied in order to understand the behaviour of the network with training and tests in the same view and with training and tests in different views. In doing so we aim to understand if the network can handle accurately new views without being trained for these cases.

In the left view of session 1 (6.8), five tests with the same data were made with different data parts. The data was divided into five equal parts and one part

Table 6.8 Results from each fold on Session 1 left view.

Fold	MSE	MAE
1	73.5	4.05
2	37.35	4.01
3	110.5	5.92
4	47.2	4.41
5	102.68	4.45

Table 6.9 Results obtained from different views and different optical flow for the Session 1 on left view.

Optical Flow	Session 1			MAE	MSE
	Left	Center	Right		
Raft	X			4.53	71.59
	X		X	3.82	78.68
	X	X	X	4.20	75.82
Farneback	X			3.62	85.11

of the five was used for testing and the others for network training. Each training corresponds to a fold, we can see that the dataset for session 1 is not homogeneous, this can happen when different parts of the video have more or fewer cars passing. If there were fewer cars passing the network would not have the needed information to make a good train, consequently, the test results would be worst.

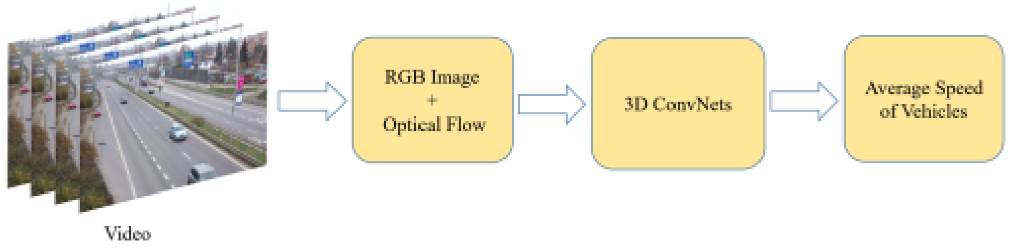


Figure 6.6: Flow chart example of the end-to-end network.

The next tests were made with just one fold for each view combination because the training of the network was too heavy, each view took about 72 hours. We tested different validations with different views and optical flows, the results are shown in Table 6.9.

The errors in the table with the reduced dataset training gave interesting information about what we could accomplish by using the complete dataset. In doing so it would increase the accuracy and at the same time a more general network could get speed estimations on any view.

7 Conclusions

Content

The purpose of this project was to present optimized solutions for speed estimation to Brisa, this solutions are based in deep learning and CNN technologies. The main issue was solve a problem where a single camera can estimate speed of cars passing in a road without the need of human intervention.

Two methods were addressed in this dissertation, tracking by detection and an end to end network. The first was studied and tested in order to present the best results in real-time, leading to a system able to estimate the vehicles speed in real time accurately. The second approach aims to acquire information on the use of 3d convolution networks in speed estimation.

For the speed estimation with object detection and tracking we discussed the different steps required to tackle this problem. On the subject of the calibration, 2 different calibrations were used. One of the calibration parameters was taken from the ground truth dataset and it was used to validate the detection and tracking systems. To accomplish the main objective - automatically estimate vehicle velocity - it was used an already implemented calibration system from ISR.

In the tracking by detection system, the main problem was to create a convolution neural network that would be able to detect vehicles on a video in real-time ($>$15 fps). After testing the existing state-of-the-art methods we ended up using the RPN with embedding. This method gives more information to the network while the network is detecting objects. It can have higher errors when objects are out of the proposal region, reducing its accuracy. Nevertheless it can work with high accuracy without loss of performance when trained for each type of circumstance. The DeepSort algorithm showed lower false-positive objects and was able to do this in real-time, thus maintaining a real-time system.

This system is prone to errors since it is composed of 3 different components, each one developed to give a solution for the 3 problems calibration, detection and tracking. Nevertheless, by increasing the number of components in a system were also increasing its errors. To solve this problem, an end-to-end method was proposed in order to understand if it can be used to estimate speed with an overall system created to tackle all the problems at once relying on Artificial Intelligence (AI).

The end-to-end network is a 3D network that instead of differentiating each problem in speed estimation, it uses a cluster of images, 16 frames each, that outputs the mean speed estimation of the objects in the video. This approach reduced the variability of the system since it had less separate components. The trained network showed good results with 1/6 of the supposed training. As known, the more data the training has, the better results a network can have. With the results shown and with less training the network could estimate speed with lower error. This makes the training of the network with more results reliable. The results obtained are a step forward on the use of this type of networks to make speed estimations through computer vision.

7.1 Future Work

Although some interesting results were achieved with both systems, there where high peek errors that should be mitigate. On the tracking by detection network it could have been implemented a detector with better detection precision (reducing miss detection) or with higher embedding features (increasing tracking association), nonetheless it would reduce its performance not allowing to run it in real-time. The End-to-end network could be trained with higher amount of data to reduce its errors. Another interesting suggestion for future work, would be to implement the end-to-end network with reduced time sequences, this could increase specific car features, reducing it speed estimation error.

Bibliography

[1] M. T. et al., “ Traffic Flow Analysis with Multiple Adaptive Vehicle Detectors and Velocity Estimation with Landmark-Based Scanlines.” *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 100–1007, 2018.

[2] H. X. A. Z. Z. Tang, G. Wang and J. Hwang., “ Single-Camera and Inter-Camera Vehicle Tracking and 3D Speed Estimation Based on Fusion of Visual and Semantic Features.” *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 108–1087, 2018.

[3] M. O. . R. H. Rajab, Samer Al Kalaa, “Classification and speed estimation of vehicles via tire detection using single-element piezoelectric sensor,” *Journal of advanced transportation*, vol. 50, 2016.

[4] W. G. N. E. A. Abdelgawad, A. El Mahdy and A. Shoukry, “Estimating Vehicle Speed on Highway Roads from Smartphone Sensors Using Deep Learning Models,” *IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 979–986, 2019.

[5] J. e. a. Sochor, “ Traffic surveillance camera calibration by 3D model bounding box alignment for accurate vehicle speed measurement.” *Comput. Vis. Image Underst.*, vol. 161, pp. 87–98, 2017.

[6] J. S. et al., “ Comprehensive Data Set for Automatic Single Camera Visual Speed Measurement.” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1633–1643, 2019.

[7] B. T. N. D. C. Luvizon and R. Minetto, “ Vehicle Tracking and Speed Estimation from Traffic Videos.” *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 153–1537, 2018.

[8] T. T. T. N. M. C. H. N. Phan, L. H. Pham and S. V. Ha, " A Real-time Vehicle Detection for Traffic Surveillance System Using a Neural Decision Tree." *25th Asia-Pacific Conference on Communications (APCC)*, pp. 256–261, 2019.

[9] M. . F. C. Kampelmühler, Moritz Müller, "Camera-based vehicle velocity estimation from monocular video," 2018.

[10] Mei and H. Ling, "Robust Visual Tracking and Vehicle Classification via Sparse Representation," *EEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2259–2272, 2011.

[11] A. Einstein, "Vehicle speed estimation by license plate detection and tracking," *EEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6563–6567, 2014.

[12] P. M. B. Silva and J. Batista, "Vehicle Occupancy Detection for HOV/HOT Lanes Enforcement," *IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 311–318, 2019.

[13] K. I. S. Kamijo, Y. Matsushita and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 2, pp. 108–118, 2000.

[14] S. H. M. A. S. M. Sun S, Akhtar N, "Deep Affinity Network for Multiple Object Tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[15] X. H. et al., " SINet: A Scale-Insensitive Convolutional Neural Network for Fast Vehicle Detection." *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1010–1019, 2019.

[16] M. Y. Z. Dong, H.; Wen, "Vehicle Speed Estimation Based on 3D ConvNets and Non-Local Blocks.. [On the electrodynamics of moving bodies]," *Future Internet 2019*.

[17] Brisa "Via Verde".

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting,"

Journal of Machine Learning Research, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[19] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," 01 2008.

[20] M. Dubská, A. Herout, R. Juránek, and J. Sochor, "Fully automatic roadside camera calibration for traffic surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1162–1171, 2015.

[21] W. L. P. D. J. C. A. Kumar, P. Khorramshahi and R. Chellappa, " A Semi-Automatic 2D Solution for Vehicle Speed Estimation from Monocular Videos." *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 137–1377, 2018.

[22] Y. . S. H. Lu, Shengnan Wang, " A high accurate vehicle speed estimation method. Soft Computing," *Soft Computing*, vol. 24, 2019.

[23] T. Huang, " Traffic Speed Estimation from Surveillance Video Data: For the 2nd NVIDIA AI City Challenge Track 1." *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 161–1614, 2018.

[24] U. U. . A. R. . M. Z. A. S. A Rahim, Hasliza Sheikh, " Vehicle velocity estimation for traffic surveillance system." *World Academy of Science, Engineering and Technology*, vol. 70, pp. 772–775, 2010.

[25] H. . W. C. . S. Biswas, Debojit Su, "Speed Estimation of Multiple Moving Objects from a Moving UAV Platform," *International Journal of Geo-Information*, vol. 259, 2019.

[26] S. P. e. a. Indu, "Vehicle Tracking and Speed Estimation using Optical Flow Method."

[27] H. Yang and S. Qu, " Real-time vehicle detection and counting in complex traffic scenes using background subtraction model with low-rank decomposition." *IET Intelligent Transport Systems*, vol. 12, no. 1, pp. 75–85, 2018.

[28] Z. Wang, L. Zheng, Y. Liu, and S. Wang, "Towards real-time multi-object tracking," *CoRR*, vol. abs/1909.12605, 2019. [Online]. Available: http://arxiv.org/abs/1909.12605

[29] S. Usmankhujaev, S. Baydadaev, and K. J. Woo, " Deep Learning Based Wrong Direction Detection."

[30] L. H. L. H. e. a. Song, H., " Vision-based vehicle detection and counting system using deep learning in highway scenes." *Eur. Transp. Res. Rev.*, vol. 11, no. 51, 2019.

[31] R. O. G. Oltean, C. Florea and V. Oltean, " Towards Real Time Vehicle Counting using YOLO-Tiny and Fast Motion Estimation." *IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 240–243, 2019.

[32] S. Chen and W. Lin, "Embedded System Real-Time Vehicle Detection based on Improved YOLO Network." *IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pp. 1400–1403, 2019.

[33] Y. . Z. H. . G. R. . H. J.-N. Wang, Gaoang Wang, "Exploit the Connectivity: Multi-Object Tracking with TrackletNet." 2018.

[34] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[35] J. Shi and C. Tomasi, "Good features to track," *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[36] C. Tomasi and T. Kanade, "Detection and tracking of point features," International Journal of Computer Vision, Tech. Rep., 1991.

[37] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *CoRR*, vol. abs/1703.07402, 2017. [Online]. Available: http://arxiv.org/abs/1703.07402

[38] M. Dubská, A. Herout, and J. Sochor, "Automatic camera calibration for traffic understanding," in *BMVC*, 2014.

[39] B. Coll and J.-M. Morel, "A non-local algorithm for image denoising," vol. 2, 07 2005, pp. 60– 65 vol. 2.

[40] Z. Teed and J. Deng, "RAFT: recurrent all-pairs field transforms for optical flow," *CoRR*, vol. abs/2003.12039, 2020. [Online]. Available: https://arxiv.org/abs/2003.12039

[41] A. Einstein, "Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]," *Annalen der Physik*, vol. 322, no. 10, pp. 891–921, 1905.

[42] S. T. L. T. R. T. F. H. Gioele Ciaparrone, Francisco Luque Sánchez, "Deep learning in video multi-object tracking: A survey."

[43] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," *CoRR*, vol. abs/1705.07115, 2017. [Online]. Available: http://arxiv.org/abs/1705.07115

[44] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[45] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015. [Online]. Available: http://arxiv.org/abs/1503.03832

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[47] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*, J. Bigun and T. Gustavsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370.