UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Design, development and evaluation of a software architecture for tracking multiple vehicles in camera networks

Emilio Gómez García
Director: Jorge García Blázquez
Supervisor: Marcos Escudero Viñolo

-MASTER THESIS-

Electronics Technology and Communications Department
Escuela Politecnica Superior
Universidad Autonoma de Madrid
June 2019

Design, development and evaluation of a software architecture for tracking multiple vehicles in camera networks

Emilio Gómez García

Director: Jorge García Blázquez
Supervisor: Marcos Escudero Viñolo

Indra Sistemas S.A.
San Fernando de Henares
Mar Egeo, 4 Poligono Industrial 1
June 2019

Resumen

El pórtico free-flow es un sistema de peaje automático que funciona gracias a la información proporcionada por las diferentes cámaras del sistema y al uso de las nuevas tecnologías. Este trabajo se centra en una parte esencial para la creación de esta infraestructura, el desarrollo del software necesario para detectar, clasificar y rastrear objetivos a través de una red de cámaras, así como el estudio del hardware necesario para su despliegue.

En primer lugar, es estudiado el estado del arte para entender los diferentes métodos que existen para el desarrollo de cada una de las tareas de este sistema. Posteriormente, se estudiará la propuesta realizada para el diseño seleccionado. Este diseño estará formado por tres cámaras situadas en el pórtico, que estarán conectadas a una placa de procesamiento de imágenes y a un foco para proporcionar la iluminación necesaria durante la noche. Además será necesario utilizar un sistema central que realizará las tareas de comunicación entre las tres cámaras, con el fin de disponer de un diseño compacto que almacene la información de cada vehículo que pasa por el pórtico. Esta información contendrá el tipo de vehículo, su matrícula y el tipo de ejes que utiliza. Posteriormente, se realizará el estudio de los sistemas de hardware que se utilizarán para la composición de este sistema multicámara, así como algunas de las secciones de software más destacadas.

Se propondrá un sistema experimental para el análisis de los resultados globales del sistema, así como la comparación entre los diferentes algoritmos propuestos, con el fin de analizar sus propiedades y determinar cuál de ellos es el mejor.

Este TFM forma parte de un proyecto real que actualmente se está desarrollando en INDRA, para su implantación tanto en carreteras nacionales como internacionales.

Palabras Clave

Pórtico de transito libre, detección de objetos, clasificación de objetos, seguimiento de objetos, reconocimiento óptico de caracteres, JetsonTX2, camara Daheng Imaging.

Abstract

The free-flow portico is an automatic toll system that works thanks to the information provided by different system cameras and the use of new technologies. This work is focused on an essential part for the creation of this infrastructure, the development of the software needed to detect, classify and track targets across a network of cameras, known as multi-target multi-camera tracking, as well as the study of the hardware necessary for its deployment.

First of all is to study the state of the art to understand the different methods that exist for the development of each of the tasks of this system. Afterwards, the proposal made for the selected design will be studied. This design will be formed by three cameras placed on the portico, which will be connected to a board of image processing and a strobe to provide the necessary lighting at night time. In addition to these systems it will be necessary to use a central system that will carry out the tasks of communication between the three cameras, in order to have a compact design that stores the information of each vehicle that goes through the portico. This information will contain the type of registration vehicle and the type of axles that it used. Later, the study of the hardware systems that will be used for the composition of this multicamera system will be carried out, and some of the most prominent software sections.

An experimental system will be proposed for the analysis of the overall results of the system, as well as the comparison between the different proposed algorithms, in order to analyze its operations and determine which one of them is the best.

This work is part of a real project that is currently being developed in INDRA, for the implementation in Spanish and international highways.

Keywords

Free-flow portico, object detection, object classification, object tracking, optical character recognition, multi-camera system, Jetson TX2, Camera Daheng Imaging.

Acknowledgements

I would like to start by thanking Marcos for his help, support and involvement with me and my project during the entire development of it. Your contribution has been essential to the realization of the same, helping me to solve the setbacks that have occurred.

Afterwards, I would like to thank all my colleagues at INDRA for their support, making my first experience in a company a very positive experience. In particular, thank my tutor Jorge for his support and help whenever I have needed it.

The realization of this master has been very positive, both personally thanks to the moments lived in Budapest and Bordeaux and professionally, opening the doors to the business world. However, the most important thing I take away are friends, but real friends, after a year of living together... you learn to love them... or not?.

Thanks to all my friends, especially Duna to perform the tasks of paparazzi, I hope that one day you learn to use a camera, and those who have helped me with language.

To conclude, of course, thank you for your support to my parents and sister, many times supporting me more than necessary. They are my source of inspiration.

Many thanks to all the family! :)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Tolls on highways have been used by the society for many years, but thanks to the development of new technologies, automatization of toll infrastructures is now possible, yielding simpler, cheaper and convenient mechanisms for drivers.

This work is focused on the setup of a new infrastructure: the free-flow portico. The free-flow portico proposes an electronic toll collection system, which eliminates barriers and charging routes, allowing the free movement of vehicles and avoiding the congestion generated by traditional toll collection processes.

The multitracking of vehicles and their classification in a camera system is an essential aspect in the creation of the free-flow portico. However, it is not the only one since the global free-flow system includes lasers as an additional method of classification, or antennas for the electronic collection system.

The continuous improvement and constant innovation of the state of the art technologies, positions the free-flow gantry at the forefront of toll collection solutions. The 'free flow' system is present on the most modern highways in countries such as Australia, Chile, Singapore or the United States (for further references see www.lavanguardia.com).

The objective of the deployment of the toll highways is to improve movement on their roads. Nevertheless, traditional systems of toll with cabins are prone to cause long queues of waiting, a counterproductive factor that hinders their utility. Carlos Frugoni, president of AUSA, told Infobae that traditional tolls had 200 cars per hour, while with the new system, 800 could pass. In addition, he estimates that accidents and pollution will be reduced thanks to the elimination of the stops and waiting times (for further references see www.infobae.com).

On toll highways, the taxes imposed on each vehicle depends on the weight of the vehicle, imposing a higher rate on heavier vehicles because the deterioration of the road is greater in its path. The weight of the vehicles depends on the type of wheel used, vehicles of greater tonnage will use double wheels to support their weight.

The weight of the vehicles and therefore the rate they must pay, depends on the type of axle they use. In Spain, according to the Ministry of Development (for further references see www.fomento.gob.es), the tariff groups are:

- Light (L). This type of vehicle does not contain axles of double type. It includes motorcycles with or without sidecar, vehicles with or without a trailer, vans with two axles (four wheels) and minibuses with two axes (four wheels).
- Heavy 1 (H1). Vehicles in this category may contain double axles. It includes trucks of two or three axles, vans and minibuses with two axles and with one double axle trailer.
- Heavy 2 (H2). This type of vehicles usually contains double axles. This group encompasses trucks of at least four axles with or without a trailer, cars, vans or minibuses of two axles, with a trailer of one or more axles with at least one of them double.

1.2 Objectives

The objective of this project is the design, development and evaluation of a software solution to detect, track and re-identify vehicles in a free-flow portico toll collection system by analysing video captured by cameras placed on the porticoes and on the sides of the road.

This objective is divided into the following partial objectives:

1. Study of the state of the art to become familiar with the techniques of detecting, classifying, identifying and tracking objects.
2. The development of software algorithms, implementation of the software necessary for vehicles and axles on a PC and a Jetson board (for further references see www.nvidia.com).
3. The development of a centralization system, to carry out vehicle tracking and to manage communication between the cameras of the system.
4. Study of the hardware systems used to design the portico.

5. Experimental evaluation discussion, and comparison of the results obtained by the different algorithms integrated.

1.3 Document organization

The rest of the document is arranged into 8 sections

1. Introduction. Work motivation and objectives.
2. State of the Art. State of the art about convolutional neuronal networks (CNNs), object detection, object classification, object tracking, re-identification (ReID) and multi-camera tracking (MTMC) intended for use in vehicles field.
3. Proposed System. Description of the general design of the device and each of the blocks that compose it. The design developed both for each of the cameras and for the central system will be explained.
4. Development. Description of the hardware used for the system, and of the communication between the different devices and with the central system from which the entire network is to be monitored. This chapter also deals with some important features of the software.
5. Experimental evaluation. Evaluation of the results obtained with the proposed system comparing the results obtained with the different algorithms developed or integrated.
6. Conclusions and future work. Overall summary, personal reasoning and potential improvements that could be introduced in this system.
7. Bibliography.

Chapter 2

State of the Art

2.1 Introduction

The opportunity of highway tolls to use traffic cameras to track vehicles over large areas that span multiple cameras and the increasing interest in deep learning techniques, such as Convolutional Neuronal Networks (CNNs), brings great advantages over traditional toll methods.

This goal has to address three different but closely related research problems:

- Detection, classification and tracking of targets within a single camera, known as multi-target single-camera (MTSC) tracking.
- Re-identification of targets across multiple cameras, hereinafter ReID.
- Detection and tracking of targets across a network of cameras, known as multi-target multi-camera (MTMC) tracking. MTMC tracking can be regarded as the combination of MTSC tracking within cameras and image-based ReID with spatiotemporal information to connect target trajectories between cameras [1]. Much attention has been paid in recent years to the problem of vehicle-based re-identification and MTMC tracking.

Object detection and classification play an important role in the computer vision field. To achieve good results, both object detection and classification needs pre-processing steps, such as eliminate noise, adjusting contrast, re-sizing, and background subtraction. Different algorithms such as Scale-Invariant Feature Transform (SIFT) [2], Speeded-Up Robust Features (SURF) [3] or Histogram of Oriented Gradients (HOG) [4] are used to extract features and points of interest from images. For a long time, object detection and classification depended on these hand-crafted features. Consequently, selecting good features is crucial to achieving high accuracy [5].

However, lately, interest in deep learning techniques, such as Convolutional Neuronal Networks (CNNs), has increased, although the first CNN dates back to the 1990s [6]. Overall, CNNs have achieved outstanding results not only within the computer vision field but also within the audio field. The advantages of these networks, come from needing no extensive pre-processing [7] and hand-crafted features.

Sermanet et al. [8] development of a network that can be used both to detect and to classify. Initially, the CNNs were only used to classify, but here is proposed some modifications on the AlexNet layers to be able to classify, like including non-overlapping pooling, in the first two layers. The detection model is trained to generate the bounding boxes that represent the position of the detected objects. Detection and classification CNNs are ran at the same time, and the final result will be the union of both outputs.

Vehicle tracking systems obtains position information from a vehicle detected over time. Tracking is needed when analyzing traffic flow and behavior or identifying traffic accidents [9]. There are various factors that may turn the recognition process needed to track more difficult, such as: variation of lighting and weather conditions, the number of different vehicle types or the distance between the vehicles and the capture device.

The fields of detection [10], classification [11] and tracking [12] of vehicles have been exhaustively studied, whereas research on ReID has been scarcer. The objective of ReID is to find in a database the images of the same vehicle captured by different cameras. This task is essential in different fields of the artificial vision including: video intelligent transportation [9], surveillance [13] and urban computing [14]. The most precise way to perform this ReID is by registration of the car's license code.

2.2 Convolutional Neuronal Networks (CNNs)

It is not the objective of this master's degree project to detail the operation of the CNNs, we here use the CNNs under a black-box paradigm, i.e. we understand the CNNs as a type of neural networks able to detect and classify an entry data providing an exit, after a training previous made from a large amount of data. They are formed by neurons interconnected which in turn form layers, the number of these layers being the depth of the network. These neurons have as input neurons of the layer before and its output feeds one or more neurons of the next layer. Each of these neurons has an associated weight that is obtained from the training of the network. In short, a trained network is defined by its architecture (organization of neurons and their connections) and by the weights learned or adjusted during the training.

The particularity of convolutional networks is that these weights are organized in multi-dimensional filters (kernels) that are applied locally on multi-dimensional data at the entrance of each convolutional layer. In addition to the convolutional layers, there are additional layers within the CNN architectures, in particular, some types of additional layers that use the networks analyzed in this work are:

- Fully-connected, where all the neurons of a layer are connected to those of the adjoining layer.
- Pooling layers to reduce the size of the input data and enlarge the receptive field of deeper layers.
- Rectified Linear Unit or ReLU to increase the non-linearity of the data, making the output equal to the input if it is positive and 0 otherwise.
- Softmax, last layer that adapts the result to convey probability-like information.

In the following sections, we study some interesting CNNs for the development of the proposed system.

2.2.1 MobileNet V1

The main objective of MobileNet Version 1 (V1) [15] is to replace the convolutional layers needed for this type of networks with others that have a smaller computational expense and that obtain similar results. These layers are known as depthwise separable convolution blocks.

These layers are arranged in two sub-layers. First, the input filter with a depthwise convolution layer, and then, the combination of these filtered values with a pointwise convolution layer creating the final result of the depthwise separable convolution block (see Fig:2.1).

This depthwise separable convolution block formed by pointwise and depthwise layers, performs the same function as the convolutional layers in less time.

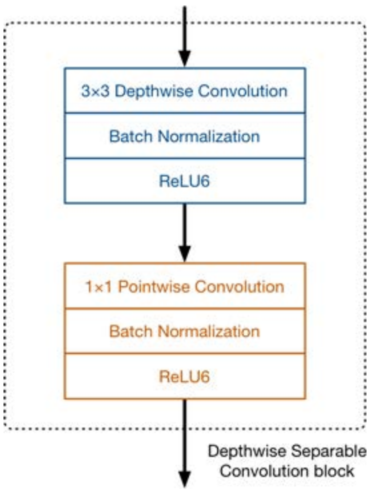

3×3 Depthwise Convolution
Batch Normalization
ReLU6
1×1 Pointwise Convolution
Batch Normalization
ReLU6
Depthwise Separable Convolution block

Figure 2.1: Depthwise separable convolutions

The MobileNet V1 architecture [15], is formed by a 3x3 convolutional network layer as the first step, followed by 13 depthwise separable convolution block layers as seen above.

Reduction in size in some of these depthwise separable convolution block, is not due to the use of pooling layers, but to the use of 4 depthwise layers with a stride of 2. The introduction of this stride, causes the size reduction of the next data layer and increment the output channels. If the input image has a size of 300x300x3 then the output of the last depthwise separable convolution block will be a 10x10x1024 feature map (see Fig:2.2).

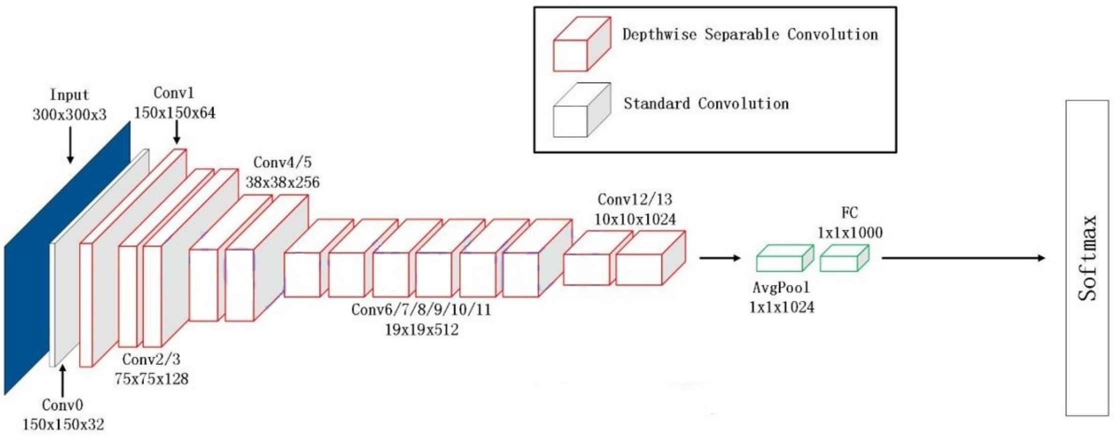


Figure 2.2: Mobilenet V1 scheme architecture.

Convolution layers of this architecture are followed by normalization. Specifically, this network uses as activation function a rectified linear unit (ReLU). Particularly, ReLU6 is used, which is a more robust variation of ReLU and its shape is similar to a sigmoid (see Fig:2.3).

$$y = min(max(0, x), 6) \tag{2.1}$$

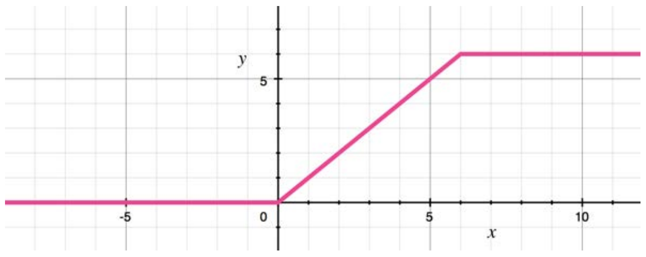


Figure 2.3: ReLU6 scheme.

When this structure is used to classify, an average pooling layer is introduced at the end, followed by a fully-connected classification layer and a softmax.

This network achieves a speed 9 times higher than other networks with the same accuracy. This is due to the use of new depthwise layers replacing the convolutional ones.

2.2.2 Single Shot Multibox Detection (SSD)

To know the meaning of SSD, let's first see the meaning of each one of the words that give them the name:

- Single Shot, implies that both the detection and the classification tasks are carried out within a single forward pass over the network.

- MultiBox, it is a technique implemented by Szegedy et al. [16] for bounding box regression.

- Detector, cause the network is focused on detect objects.

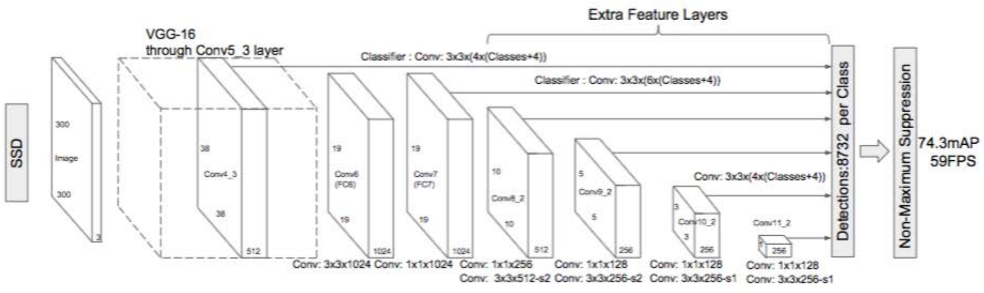

Extra Feature Layers
VGG-16 through Conv5_3 layer
SSD
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
Conv: 3x3x(4x(Classes+4))
Detections:8732 per Class
Non-Maximum Suppression
74.3mAP 59FPS
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256 Conv: 3x3x512-s2
Conv: 1x1x128 Conv: 3x3x256-s2
Conv: 1x1x128 Conv: 3x3x256-s1
Conv: 1x1x128 Conv: 3x3x256-s1

Figure 2.4: Single Shot Multibox Detection (SSD) architecture.

The SSD method (see Fig:2.4) is a feed-forward CNN which output is the position of the detected objects (represented by bounding boxes) and the scores showing the belongingness to particular classes of objects. This method uses a non-maximum suppression step to obtain the definitive results.

The SSD object detection architecture is composed of two parts. A feature maps extraction part using VGG-16 [17] or MobileNet, and an object detection part on top of it.

Output bounding boxes are restricted to a set of scales per feature map location in the image. On the detections of interest adjustments are made to the boxes to match better the object's shape, combining predictions of multiple feature maps with different resolutions to handle objects of different sizes.

2.2.3 SSD MobileNet V1

Single Shot Multibox Detection architecture (see Fig:2.4) builds on top of other architectures, typically on MobileNet architecture (see Fig:3.4), discarding fully connected layers [4]. The choice of these architectures as the basis on which the SSD architecture is introduced, is due to its great precision in the task of classifying high-quality images.

Instead of the fully connected layers that make up the end of the MobileNet V1 architecture, some convolutional layers are added with the objective of extract features at multiple scales, decreasing the size of the input in each of those layers.

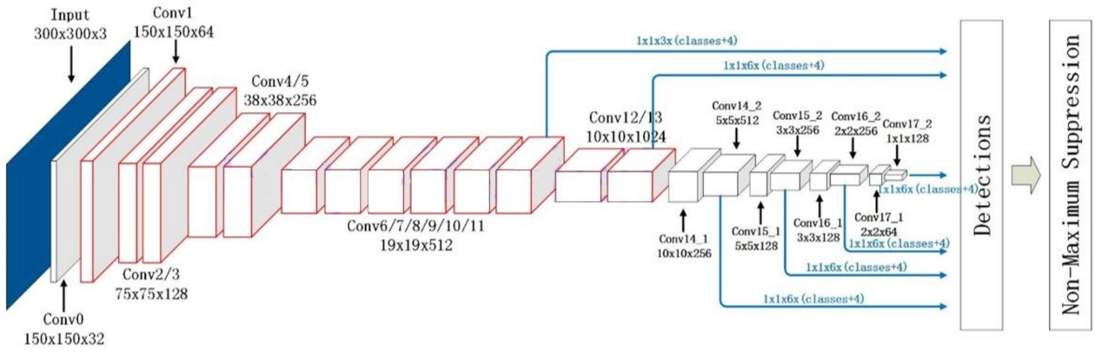


Figure 2.5: SSD MobileNet V1 scheme architecture.

2.3 Object detection

Object detection refers to the ability to locate objects in an image by identifying each object.

Detecting vehicles from videos is one of the bases for systems of intelligent traffic management [18]. In recent years many efforts have been put into the development of algorithms in this field [19] [20], and creating benchmark for evaluation [21]. In addition, the advance of CNN has provoked incredible results in the task of vehicle detection and in other detection tasks [22] [23].

In 2013 Girshick et al. [24] introduced a region CNN proposal denominated R-CNN. This method uses selective search to find possible regions in which the object is located, where the objects of interest will be found in several of those regions. Later will be obtained 4096 feature vectors with AlexNet in each proposed region. Finally, each vector will be classified using SVMs, where each SVM has been trained for a class, getting better results than many algorithms using low-level features, as shown on the ILSVRC2013 detection dataset.

Ming Liang and Xiaolin Hu [25] propose a recurrent CNN. This solution is inspired by recurrent connections in recurrent neural networks. The authors develop a recurrent convolutional layer that is used instead of regular convolutional layers in the proposed model. Only the first convolutional layer of the network is not recurrent. The base network and training process were adjusted from AlexNet. The depth of the network is increased by adding recurrent connections.

YOLO [26] and SSD MultiBox (Single Shot Detector) [27] are two of the most recent object detection approaches. Some previous works use classifiers to perform object detection. In contrast, YOLO predicts bounding box coordinates along with the class score within the same neural network. Unlike region proposal-based networks YOLO uses the whole image instead of separated regions to make decisions. The network architecture is based on GoogleLeNet [28] where inception modules are replaced by reduction layers. Twenty convolutional layers are pre-trained and then another four convolutional and two fully-connected layers are added to the model. Each input image is divided into a grid. Each grid cell predicts bounding box coordinates, the bounding box confidence and a class probability.

2.4 Object classification

Object classification algorithms detect the class that corresponds to an input object, CNNs have been a great step forward in this task.

The algorithm proposed by Krizhevsky et al. [29] in 2012, was a breakthrough in the treatment of this task with a long dataset thanks to the use of AlexNet. With this algorithm it was shown that CNNs are able to work with large amounts of images getting good results, thanks to the depth of them. Without any pre-processing step of the images, the network is formed by 5 convolutional, 3 fully-connected layers that use ReLU as activation function and a final softmax layer with 1000 units. To avoid overfitting during training they propose data augmentation and dropout.

A new VGGNet architecture [30] was introduced in 2014, which remained in first and second position in the 2014 ILSVRC Competition for classification and localization of objects respectively. This network is based on AlexNet, using smaller kernels of size 3x3 in all the convolutional layers. In addition, successive convolutional layers are used without any pooling between them. The authors tested different configurations, but the best results were obtained with the architectures composed of 16 and 19 layers.

The CNN-RNN is obtained by combining a recurrent neural network with a convolutional neural network [11]. The recurrent neural network is used to classify multiple objects from the same image. While adding RNN to the original network gives better results. Authors proposed a CNN based on VGGNet, where the RNN provides the relation between the images and their classes.

2.5 Object tracking

Object tracking systems obtains the position information from an object detected over the time that is present at the scene.

The first algorithms focused on solving multiple object tracking (MOT) tasks, was based on Multiple Hypothesis Tracking (MHT) [31] or in the use of Joint Probabilistic Data Association (JPDA) filters [32][33]. The main problem of these algorithms lies on their inability to track multiple objects in real time. Later, other proposals were made for this task, such as those made by Rezatofighi et al. [33] based on the JPDA method [32], or by Kim et al. [34]. However, these methods are also not able to solve the real time problem.

All tracking algorithms need to detect objects at some point. They keep track of the detected objects by correlating them during the time that they are on the scene. For this, features such as location or appearance are used. Models can be based on the appearance of each detected object [35] or the global appearance [12]. Some of these models also add the movement feature, with the aim of associating detections and trackers [12]. This connection between detections and trackers can also be made through other methods as with the Hungarian algorithm [36].

Our approach is based on two classical and extremely efficient methods. One is the Kalman filter [37], which predicts the position of the vehicles depending on their movement. The second one is the Hungarian method [36], which makes the association between detections and trackers. This tracking method provides great performance and reliability to carry out real-time tracking (online tracking).

2.5.1 Kalman Filter

The Kalman Filter [38][37] is a recursive optimal estimation algorithm that predicts the parameter of interest given a set of measurements.

The first step of the Kalman Filter is to obtain the model of the system. The physical system is modeled by a state vector x (see first Eq:2.2), simply called the state, and a set of equations called system model. The system model z (see second Eq:2.2) is an equation of vectors that describes the evolution of the state over time.

$$x_k = Fx_{k-1} + Bu_k + w_{k-1} \tag{2.2}$$

$$z_k = Hx_k + v_k \tag{2.3}$$

The first equation tells that any x_k is a linear combination of its previous value x_{k-1} adding a control signal u_k (usually this signal is equal to 0) and w_{k-1} a Gaussian vector to model the noise associated with the system model. Where usually the transition matrix of states F and the matrix B are constant matrices that define the model.

The second equation tells that in every instant k we have a noisy observation of the state vector. Where H is the called measurement matrix and v_k is a random vector modelling the uncertainty associated with the measurements, typically a Gaussian signal.

Once we have that the model is fit into the Kalman Filter model, the next step would be to obtain the initialization of the previous parameters. Also, to start the process it is necessary to estimate $\hat{x}_0$ and P_0 of the prediction and update equations.

Kalman filter consists of two steps [38]: prediction and update. The first step obtains the current state from previous states and the second step is used to correct the state. The system iterates between these two steps until calculating the best system estimator $\hat{x}_k^-$ [38] (see Fig:2.6).

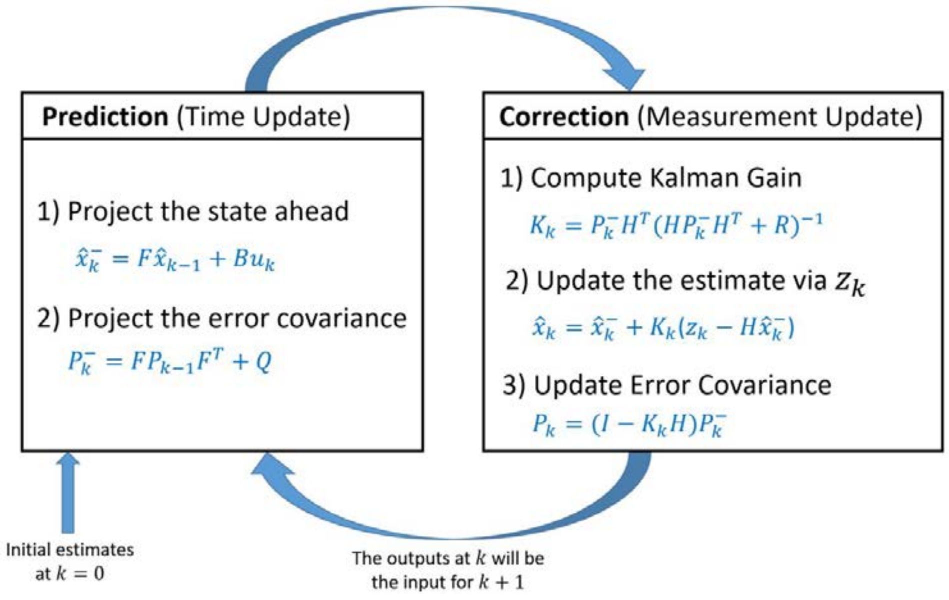

Prediction (Time Update)
1) Project the state ahead
$\hat{x}_k^- = F\hat{x}_{k-1} + Bu_k$
2) Project the error covariance
$P_k^- = FP_{k-1}F^T + Q$
Correction (Measurement Update)
1) Compute Kalman Gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
2) Update the estimate via z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
3) Update Error Covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates at $k = 0$
The outputs at k will be the input for $k + 1$

Figure 2.6: Kalman Filter iterating equations.

Where in prediction phase equations, $\hat{x}_k^-$ is the state mean, P_k^- is the state covariance, F is the state transition matrix, Q is the process covariance, B is the control function matrix and u_k is the control input.

In correction equations $\hat{x}_k$ is the estimate of x at time k. Also, P_k which is necessary for the future estimate with $\hat{x}_k$. H is the measurement function matrix, z is the measurement, R is the measurement noise covariance, K is the Kalman gain and I is the identity matrix.

2.5.2 Hungarian Algorithm

The Hungarian algorithm is an optimization algorithm which solves the problem of assigning different types of data over time, such as detections and trackers [36]. This algorithm has four steps, where the first two run only once and the last two until they find the desired optimal result. The entry of this algorithm is a positive matrix (cost matrix) of size nxn that relates the two data sets.

1. Subtract row minima. Identify the minimum of each column in the obtained matrix, and subtract it from all the elements
2. Subtract column minima. In the obtained matrix the minimum of each column is identified, and all the elements of the column are subtracted.
3. Cover all zeros with a minimum number of lines. The idea of this step is to cover all zeros in the previous matrix using the minimum number of lines possible (only vertical and horizontal lines). If all zeros can be covered with n lines, the optimal assignment will have been found and the algorithm ends.
4. Create additional zeros. Look for the smallest element in the previous matrix that has not been covered by a line. Then subtract this number from the elements that have not been covered and added to the elements covered by more than one line.

2.6 Vehicle Re-identification (ReID)

Currently, this task is still under development, with many proposed works. The first of them, based their models on the classification of vehicles by appearance, categorizing them by type and colour in a database as in the method proposed by Feris et al. [10]. Alternatively, additional information such as texture and semantic attributes has been used, as in the algorithm porposed by Liu et al. [39]. However, algorithms based on appearance are not able to re-identify the same vehicle, due to problems such as changes in illumination, occlusions or points of view. Therefore, methods capable to identify a vehicle with a unique ID have been proposed, where the license plate will be this only ID per vehicle.

The easiest way to identify a vehicle is by reading the license plate [40], as every vehicle has a different one. Automatic License Plate Recognition (ALPR) [41] are the algorithms used to recognize the plates in an image, divided into four tasks: the detection of the vehicle, the detection of the plate, character segmentation and the recognition of these characters. The Optical Character Recognition (OCR), is known as the union of the last two tasks. Actually, the OCR is one of the best processes that aims to digitize texts, these are identified automatically from symbols or characters of a given alphabet, to later be stored in the form of data.

These ALPR algorithms are widely used to identify vehicles [40] especially in controlled environments such as parking, as these methods may fail due to environmental conditions [10].

The use of convolutional neural networks also meant supposed an improvement in these algorithms. The use of siamese neuronal networks (SNN) is known for signature recognition [42], face recognition [43] or person identification [44], but currently also used in registry recognition algorithms for accurate vehicle ReID.

2.7 Multicamera tracking

Using multiple cameras with different points of view for tracking reduces the ambiguity you have with the use of a single camera, using algorithms to map the positions of an object in the different cameras to a ground plane, and later making the association of that object along the system of cameras. The different scenarios proposed by each of the cameras can be: not overlapping [45], be partially overlapping [46] or fully overlapping [47]. Our system is employed in an environment without camera overlapping.

This type of algorithms can be divided into three steps, after the definition of an operational area.

1. Object detection on each camera. Detection is obtained in the initially defined operational areas [48], where the algorithm is evaluated. With the use of these areas a controlled working environment is achieved, looking for the maximum overlap between the different cameras of the system.

2. Projection of the detections onto the operational plane. Semantic strategies are used to assign a single object to each pixel of the images.

3. Combination of detections and using back-projections to obtain the detections of the same object in the different cameras. These fusion algorithms are divided into three groups. The first method uses geometric intersections between detections to combine them. The second method is based on probabilistic algorithms, using statistical modelling for the union of detections. The last method is based on CNN, achieving good performance even with the presence of occlusions [49].

Post-processing algorithms are often used on the obtained results to improve detection localization [49].

Chapter 3

Proposed system

3.1 Introduction

This work is focused on the multitracking system for the free-flow portico. Initially, the system poses the use of three cameras in each free-flow portico (see Fig:3.1).

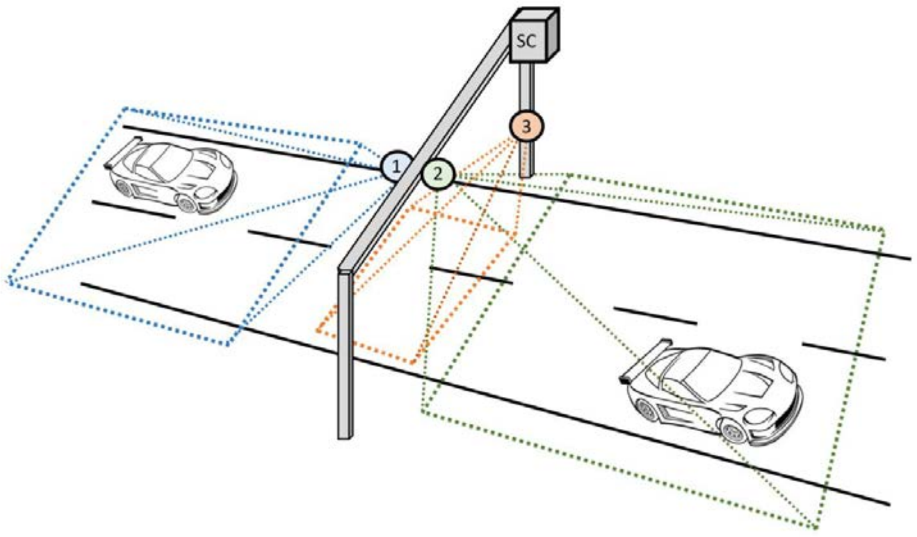


Figure 3.1: Free-flow portico infrastructure. Where 1 represents the Back camera, 2 represents the Front camera, 3 represents the Lateral camera and SC represents the centralization system.

1. Front camera to monitore and acquire front registration of vehicles.
2. Back camera to monitore and acquire rear registration of vehicles.
3. Lateral camera to axle detection of registered vehicles, differentiating between

single and double axles. Necessary because of the type of axle (simple or double) indicates if the vehicles is heavier and has to pay a higher rate [50].

The base algorithm is formed by three parallel processes to track, classify and identify vehicles and axles, merging these informations by a central system (SC) equipped with software solutions for multi-camera tracking, vehicle re-identification and camera-network management (see Fig:3.2).

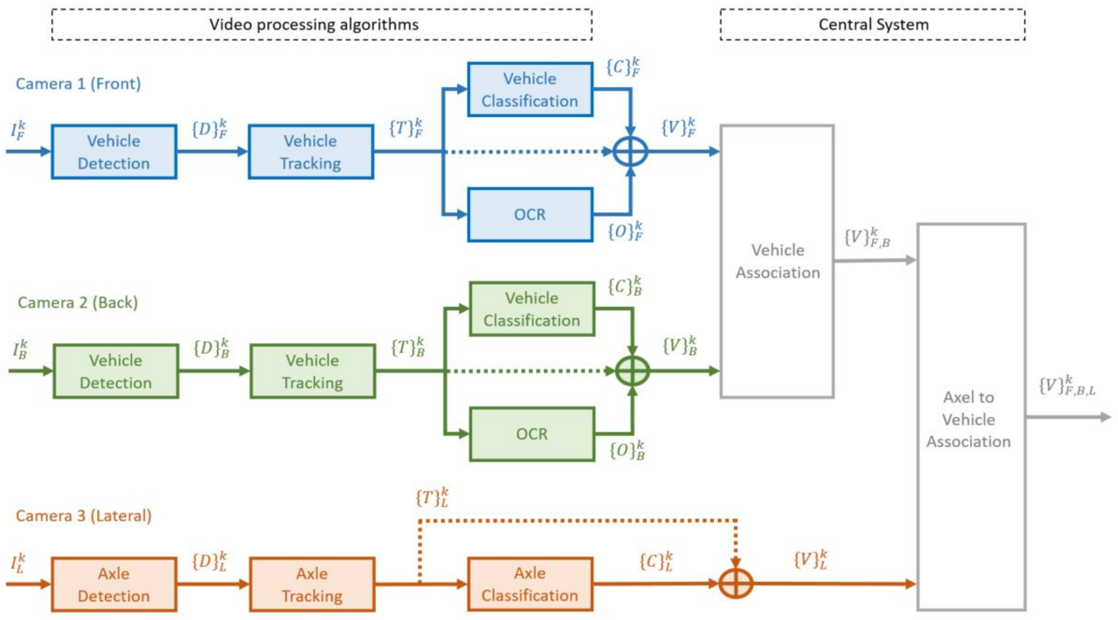


Figure 3.2: Scheme of the base algorithm. Where green boxes correspond with the algorithm process for the back camera of the vehicles, the blue boxes with the process for the front camera of the vehicles and the orange boxes with the process for the lateral camera, filming wheels. The central system is composed of three blocks.

Below is a brief explanation of each step, which are explained further in detail in each section of the chapter.

- Video processing algorithms.

  1. Input. The input for each of the three algorithms (I_B^k, I_F^k, I_L^k) that are developed in parallel (back, front and lateral) is a video frame image from the different cameras.

  2. Detection. Software to locate vehicles for camera 1 and 2, or axles for camera 3. The output is the coordinates that correspond with the bounding boxes of the detected objects ($[D]_B^k$, $[D]_F^k$, $[D]_L^k$). Where $[D]_B^k$ is a set that

contains the n detections obtained by the algorithm of the back camera in the frame k, $[D]_B^k = [d_{B,1}^k, d_{B,2}^k ... d_{B,n}^k]$.

3. Tracking. Software to keep real-time tabs on the position of each object of interest, assign to each vehicle/axle an identification number or ID to each vehicle/axle. The outputs are the tackers ($[T]_B^k$, $[T]_F^k$, $[T]_L^k$) that contain the position information and a representative ID of each tracker. Where $[T]_B^k$ is a set that contains the n trackers (position and ID) obtained by the algorithm of the back camera in the frame k, $[T]_B^k = [t_{B,1}^k, t_{B,2}^k ... t_{B,n}^k]$.
4. Classification. Software to detect the type of class that corresponds to each input object identified by its bounding box ; car, motorcycle, truck or bus for vehicle classification, and simple or double for axle classification. The outputs have the class information for each input object ($[C]_B^k$, $[C]_F^k$, $[C]_L^k$). Where $[C]_B^k$ is a set that contains the n classes obtained by the algorithm of the back camera in the frame k for the n input trackers, $[C]_B^k = [c_{B,1}^k, c_{B,2}^k ... c_{B,n}^k]$.
5. Plate recognition with OCR. Software for license identification. The outputs have the plate information for each tracker ($[O]_B^k$, $[O]_F^k$, $[O]_L^k$). Where $[O]_B^k$ is a set that contains the n plates obtained by the algorithm of the back camera in the frame k for the n input trackers, $[O]_B^k = [o_{B,1}^k, o_{B,2}^k ... o_{B,n}^k]$.
6. The final outputs ($[V]_B^k$,$[V]_F^k$,$[V]_L^k$), contain the information of time, ID, position and class for axle algorithm, adding the number of plate for the vehicles algorithms: $V_L^k = [[T]_L^k, [C]_L^k, t]$, $V_F^k = [[T]_F^k, [C]_F^k, [O]_F^k, t]$ and $V_B^k = [[T]_B^k, [C]_B^k, [O]_B^k, t]$.

- Central System.

  1. Vehicle association. The input for each frame ($[V]_B^k$,$[V]_F^k$) contains the time, ID, class and plate values of the front camera vehicle's trackers V_F^k. Comparing the number of plate of the vehicles in the back camera with the plates of the vehicles in the front camera, they are assigned the ID and the appropriate class to the vehicle matched. The time information when the vehicle goes out of the front camera and it is matched in the back camera is saved in the output with the ID, class and plate $[V]_{B,F}^k$.
  2. Axle to vehicle association. Assign the axle type $[V]_L^k$ to the vehicle information $[V]_{B,F}^k$ (ID, class and plate) using the time information, obtaining the final output $[V]_{B,F,L}^k$ a set with the information of ID, class, plate and number and type of axles, of all the vehicles detected in the time k.

In the following sections the camera algorithms are analyzed by blocks (detection, classification, tracking and OCR). Then, the central system is studied.

3.2 Object detection

Object detection refers to the ability to locate objects in an image by identifying each object, in this case, two algorithms have been implemented, one to detect vehicles and the other to detect axles.

3.2.1 Vehicle detection

Vehicle detection takes a captured image as input and produces the bounding boxes of the vehicles detected as the output. This step was implemented using two different methods, background subtraction (FGBG) and convolutional neuronal networks (CNN).

1.- Background Subtraction

The subtraction process is divided into different steps (see Fig:3.3).

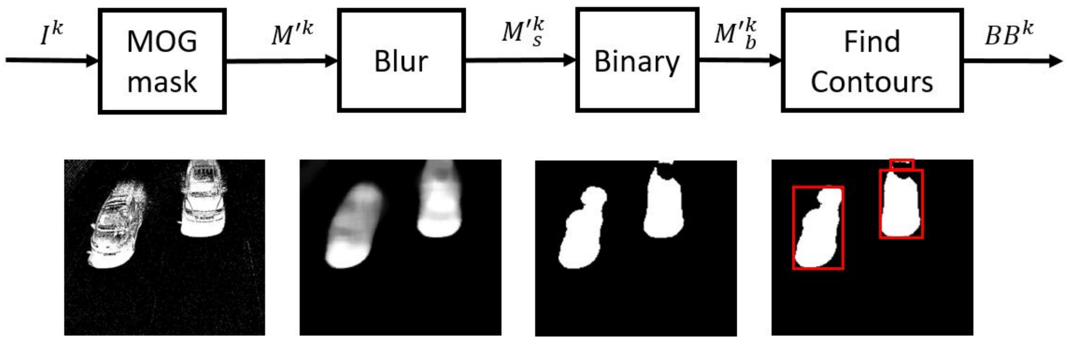


Figure 3.3: Scheme of the background subtraction method to vehicle detection. Where M' indicates that the image M is resized.

The input in the frame k of the video is the original frame in this time I^k.

To obtain the vehicles detections it is proposed the Gaussian Mixture-based Background/Foreground Segmentation Algorithm. The oriented gradient histogram is a method to detect people based on a model of shape and appearance characteristics defined by the intensity gradients and their edge distributions.

First, using the method based on [51][52] a background subtract object its created M^k applying an algorithm using Gaussian mixture probability density, where parameters of the Gaussian model are updated by using recursive equations, selecting the number of Gaussian distributions for each pixel. This method detects objects of interest as well as their shadows, and it provides good adaptability to varying scenes due to illumination changes etc.

After obtaining the foreground mask of a frame, there are some image processing steps implemented to improve the detections.

First of all, the mask M^k is resized M'^kin order to obtain a higher processing speed, since for the next steps high quality is not necessary to obtain good performance. The value of the ratio between this new image and the original one is saved to return the results to their original size afterwards.

An image blurring or smoothing step over the mask $M_s'^k$ is implemented, to make the positive values larger and more compact, and eliminate image noise like the Salt and Pepper Noise produced by the acquisition methods.

Then a thresholding step $M_b'^k$ is used to classify pixel either as foreground or background, using a threshold value of 128. Note that pixel values are in the range [0,255]. Only pixels values greater than this threshold are classified as foreground. Over this thresholded image, the contours of each object detected are obtained BB^k.

2.- Convolutional Neuronal Networks (CNN)

The name of the CNN used for vehicle detection is "ssd mobilenet v1 coco". It is an Object Detection API which contains detection models of objects trained in the COCO dataset.

The architecture of this network is a MobileNet V1. Specifically, this network used a lightweight model by introducing a Single Shot Multibox Detection (see Fig:3.4).

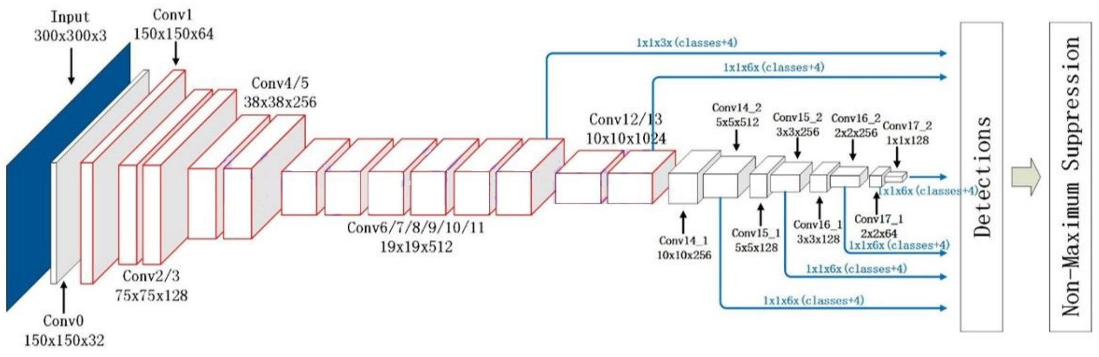


Figure 3.4: SSD MobileNet V1 scheme architecture.

This network was trained with the COCO dataset (see Fig:3.5), which is formed by images belonging to 90 different classes, of which the majority are not interesting for this project. Though this is a general-purpose detection model (not specifically optimized for vehicle detection), this model offers a good trade-off between bounding box accuracy and inference time.

In this algorithm, only the next four categories are selected to filter the detections of interest:

- Class number 3 which corresponds to Car.
- Class number 4 which corresponds to Motorcycle.
- Class number 6 which corresponds to Bus.
- Class number 8 which corresponds to Truck.

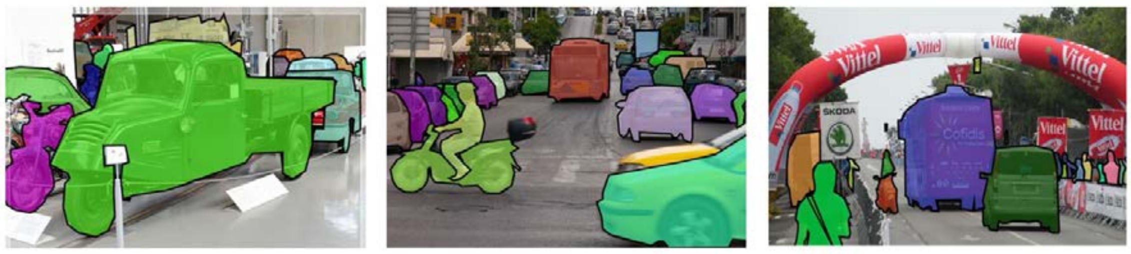


Figure 3.5: Example of the labeled image that contains the COCO dataset for car, motorcycle, truck and bus classes (for further references see www.cocodataset.org)

First, CNN runs over a frame of the image to get the detections and its scores, deleting detections that do not correspond with one of the four previous classes. Once the detections are obtained, a post processing detection step is needed, where detections are filtered by the score of each of them. A new system parameter is created 'THscore', a threshold to stipulate the necessary reliability for a detection to exist, detections with a lower score are deleted.

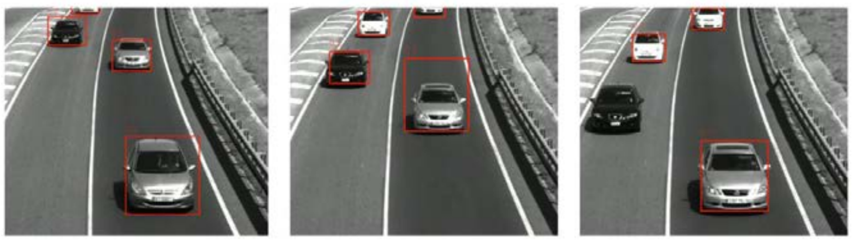


Figure 3.6: Example of detections with CNN. The images correspond with different frames of the same video using a score threshold of 0.2.

3.2.2 Axle detection

For axle detection, only the convolutional neuronal network solution works. The background subtraction algorithm obtains the foreground objects over the background, in the previous case this works because in this type of traffic videos the foreground are the vehicles and the background is the road. But when detecting axles, the objects of interest are the axles, and these are only a part of moving vehicles, therefore they can not be isolated from them by background subtraction solutions.

For the axle detection network, the original vehicle detection CNN called "ssd mobilenet v1 coco" was retrained on a dataset that collected the axels obtained from a video (see Fig:3.7).

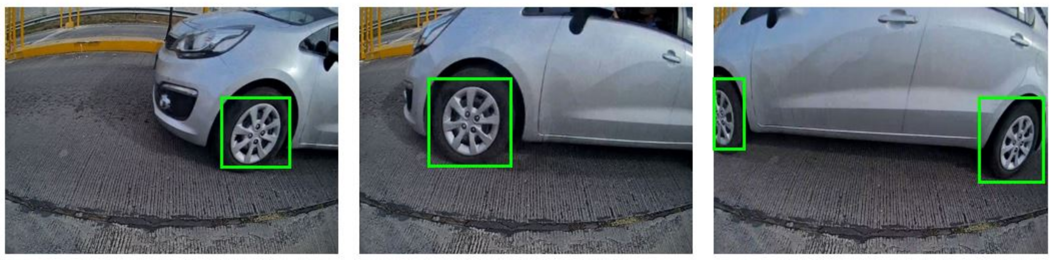


Figure 3.7: Example of frames of the video used to retrain the original CNN called "ssd mobilenet v1 coco" with the manual location of the axles.

To reduce training time without sacrificing accuracy, this CNN retraining used Transfer Learning, which is a method that allows us to use networks that have been pre-trained in a large data set, like "ssd mobilenet v1 coco". By keeping the first layers and only training the newly added layers, one can take advantage of the knowledge acquired by the pre-trained algorithm and use it for a different application.

This retraining process is divided into different steps:

1. Creating the dataset. The axle dataset was obtained from the previous video (see Fig:3.7) with the manual location of the axles in all the video frames. The final dataset is formed by different axle types (double and simple) from different vehicles and with different points of view of each vehicle axle (see Fig:3.8). This axle dataset is composed of a total of 1.145 axle images.

2. Training the model with the dataset obtained.

3. Exporting the model so it can be run in the video processing algorithm codes. This model has the bounding box detection output with its scores.

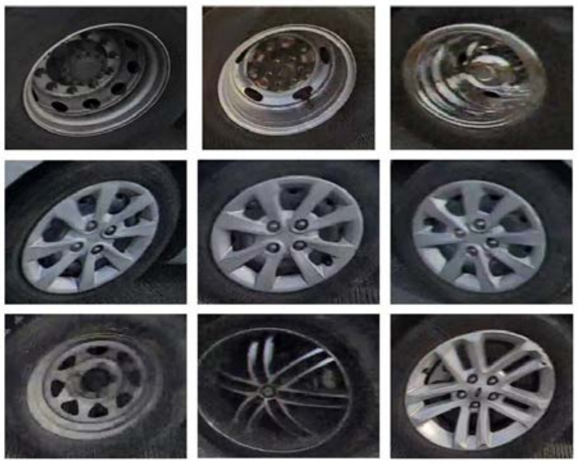


Figure 3.8: Subset of the axle images dataset. The first row corresponds with different double axles. The second row corresponds with a different point of view of the same simple axle. The third row corresponds with different simple axles.

3.2.3 Post-processing steps

After obtaining the detection of vehicles and axles through their bounding boxes localizations, two post-processing stages to refine detection are performed, regardless of the type of detection implemented.

1.- Filtering detections by the size

As the environment of implementation of these algorithms is known, a road where the vehicles/axles of interest never have a very high or very small size, filtering the detections by their size is performed (see Fig:3.9) to eliminate possible detection errors that never could be a vehicle/axle.

First, all the detections with size higher than the half of the image size, are eliminated. Then, all the 'small detections' (detections with a size lower than a new system parameter 'minArea'), in the image are eliminated because these detections correspond with objects that are not vehicles or axles, or objects of interest away from the camera that are out of the scope of the detection method.

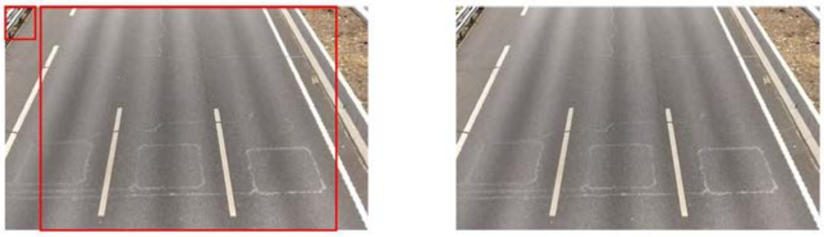


Figure 3.9: Example of filter detections by size with FGBG detection. The left image corresponds with bad detections obtained with FGBG because of a blur of the image in the capture process. Right image corresponds to the detection output after the filter by size process.

2.- Non maximum suppression

This step is implemented to prevent that there is more than one detection per vehicle. Redundant detections are eliminated by applying an intersection over the union between two detection boxes to determine if they correspond to the same detection or not. This algorithm obtains a score called intersection over union (IoU), comparing the overlap area of two detections and the area of union between them (see Fig:3.10).

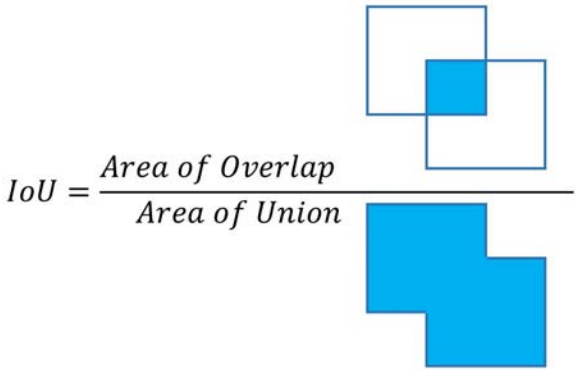


Figure 3.10: Equation and graphic example of the IoU calculation between two bounding boxes.

For this step, a new system parameter 'THiou' is created to determine the degree of overlapping to consider the detections. If the IoU score is higher than the threshold, it means that detections are a overlapped, so these detections are replaced by one that is a box over the union of both (see Fig:3.11).

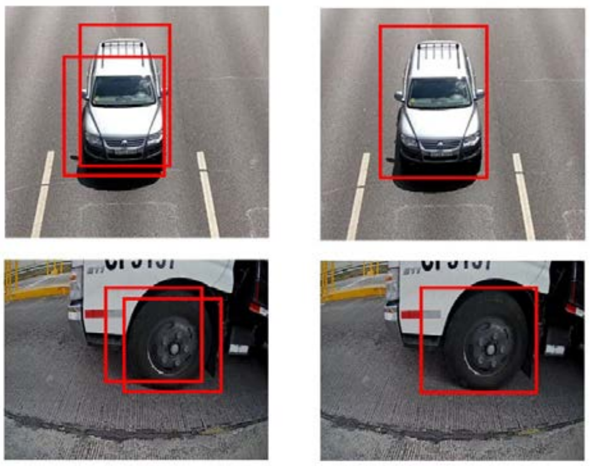


Figure 3.11: Example of how IoU works. The images in the left correspond to detections before the application of IoU. The images on the right correspond to the detections output after IoU with a threshold of 0.5.

3.3 Object tracking

The object tracking algorithm is the same in the three different video processing paths (front, back and lateral cameras), and therefore, it is independent of the object to track, axle or vehicle.

Vehicle or axle tracking is done using the Kalman filter to create the trackers and the Hungarian algorithm to assign detections and trackers. Then a tracker management module is used to initialize or remove tracks (see Fig:3.12).

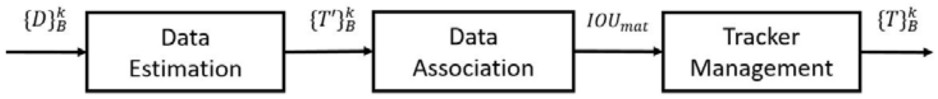


Figure 3.12: Scheme of the object tracking algorithm to the Back camera. Same for axle tracking and vehicle tracking. Data estimation step is implemented using the Kalman filter and data association step using the Hungarian algorithm.

3.3.1 Data estimation

The Kalman filter is used to create the trackers by estimating the position of the input object in the frame k at the next frame $k+1$. It is a recursive optimization algorithm that predicts the parameter of interest given a set of measurements (in this case the axle/vehicle location). For linear systems, like the highway scenario with vehicles and white Gaussian errors, Kalman filter is a good estimator.

A Kalman model assuming the constant velocity (thus no acceleration), is used to predict the position of the detected object. This position estimation of the bounding box of the detected analysis object $\hat{D}_k = (d_{up}, d_{left}, d_{down}, d_{right})$ is called tracker $\hat{T'}_{k+1} = (t_{up}, t_{left}, t_{down}, t_{right})$ (see Fig:3.13).

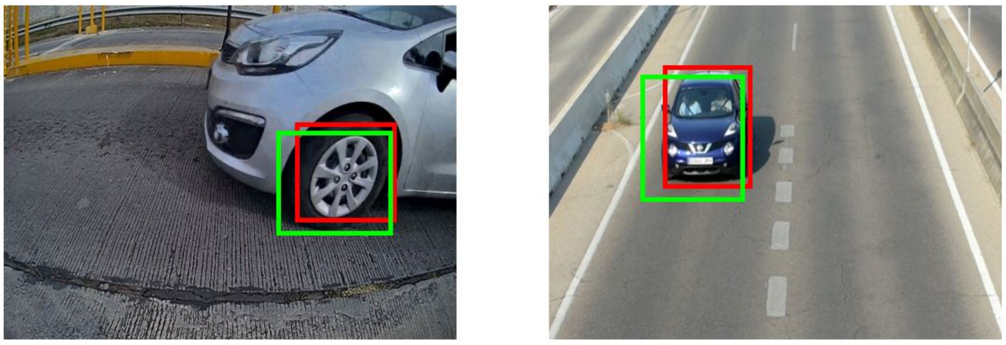


Figure 3.13: Example of position estimated by using the Kalman Filter for axle and vehicle detection. The red bounding boxes correspond to detections and the green bounding boxes with their trackers.

3.3.2 Data Association

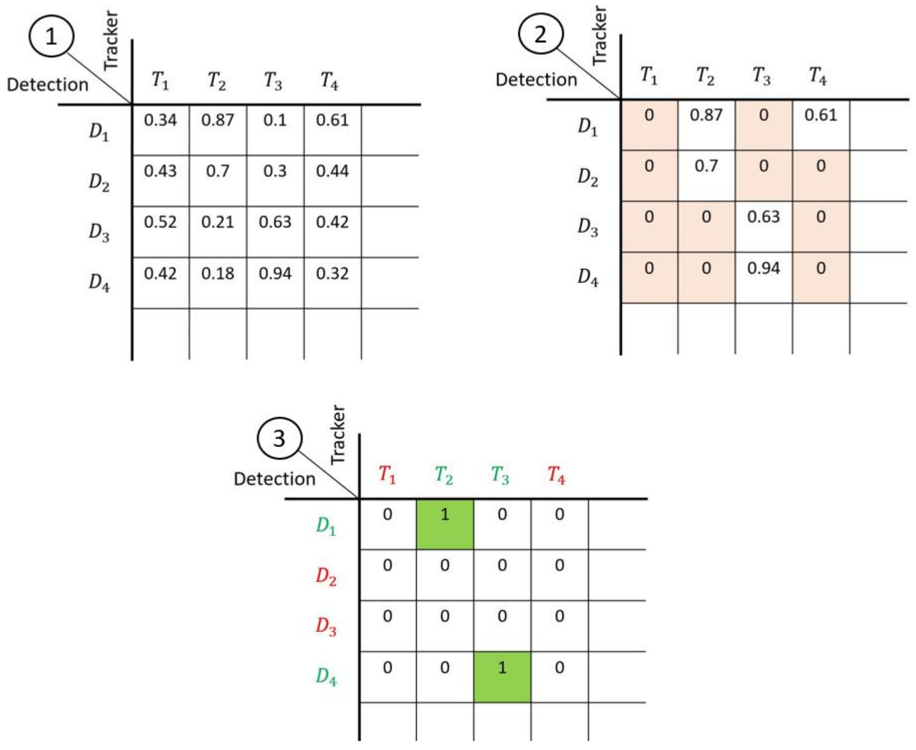
The Hungarian algorithm is an optimization to solve the problem of assigning different types of data over time (in this case, detections and trackers).

Both detections and trackers are vectors that contain the position of an object of interest (axle or vehicle) $\hat{X} = (X_{up}, X_{left}, X_{down}, X_{right})$. The Hungarian algorithm works with a matrix of numbers as input to make the assignment, we propose to use the intersection over union of a tracker and detection bounding boxes. This matrix IOU_{mat} is composed by the overlap scores between detections and trackers, obtaining them in the same way as in Section:3.2.3. These values are used as an indicator of the similarity between detections and trackers.

A new IOU overlap threshold (th_{iou2}) is employed to filter the score that is used to join trackers and detections.

The assignment problem is solved using the Hungarian algorithm to maximize the sum of IOU_{mat} after the thresholding.

Once this algorithm is finished, the IOU_{mat} indicates which pair of bounding detection-tracker boxes are matched, and which detections and trackers have not been assigned to each other (see Fig:3.14).



1

Detection \ Tracker	T_1	T_2	T_3	T_4	
D_1	0.34	0.87	0.1	0.61	
D_2	0.43	0.7	0.3	0.44	
D_3	0.52	0.21	0.63	0.42	
D_4	0.42	0.18	0.94	0.32	

2

Detection \ Tracker	T_1	T_2	T_3	T_4	
D_1	0	0.87	0	0.61	
D_2	0	0.7	0	0	
D_3	0	0	0.63	0	
D_4	0	0	0.94	0	

3

Detection \ Tracker	T_1	T_2	T_3	T_4	
D_1	0	1	0	0	
D_2	0	0	0	0	
D_3	0	0	0	0	
D_4	0	0	1	0	

Figure 3.14: Example of the update of the scores matrix. Image number 1 corresponds to the original score matrix between trackers and detections. Image 2 is the threshold of the matrix. Image 3 is the matrix after applying the Hungarian Algorithm over it, where detection and trackers in red have not been matched.

3.3.3 Tracker Management

To perform the tracking management, there are two new parameters to work with (H_{max}, H_{min}). One of them establish the number of consecutive times H_{min} that a tracker has to be matched with a detection to create a tracker and assign it an ID. The other one establish the number of consecutive times H_{max} that the tracker has not been assigned to any detection to eliminate it.

Based on the data association results, there are three scenarios for the trackers, after the data association process (see Fig:3.15).

1. Unmatched detection. When an object of interest (axle/vehicle depending on the algorithm) enters a frame and is first detected, it is not matched with any existing trackers. Thus, this detection is referred as an unmatched detection.

2. Unmatched tracker. A matching tracker with an overlap less than th_{iou2} signifies the existence of an untracked object. Also, when an object of interest leaves the frame, the previously established track has no detection to be associated with,

and the tracker is referred as unmatched tracker. If this occurs more than H_{max} consecutive times, the track is deleted.

3. Matched tracker-detection. When the same tracker is matched with a detection more than H_{min} consecutive times with an overlap higher than th_{iou2}, this tracker (T_B^k, T_F^k or T_L^k) is established as a tracker of interest and is stored in its $\hat{T}_*^k$ set being sent to the next stage of the algorithm for processing (see Fig:3.12). If there is no tracker that meets these conditions, the algorithm stops here.

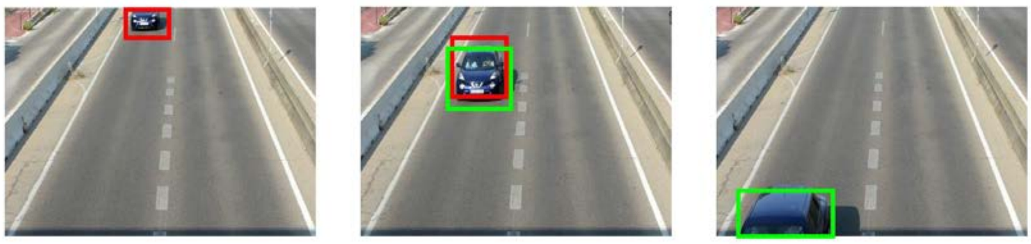


Figure 3.15: Example of position estimation for the front camera video. Starting on the left, the first image corresponds with an unmatched detection, the second one with a matched tracker-detection and the last one with an unmatched tracker.

3.4 Object classification

Object classification algorithm detects the type of class that corresponds to each input object. This algorithm takes an object previously detected as input and the outputs a class label: car, motorcycle, truck or bus for vehicle classification, and simple or double for axle classification (see in Fig:3.16). It is based on the use of Convolutional neuronal networks (CNNs), applying different training'sponer the differences of the objects that it needs to classify.

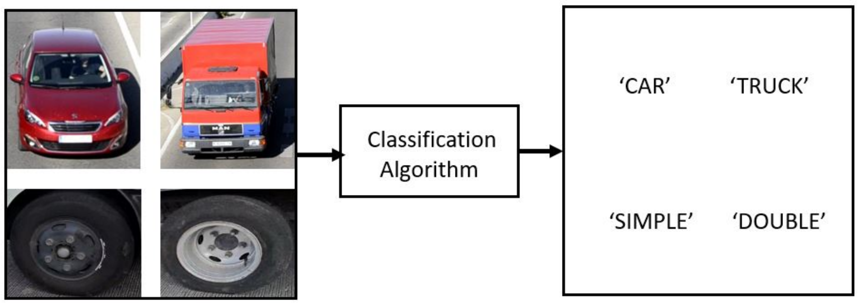


Figure 3.16: Example of classification algorithm output for different vehicles/axles input images.

3.4.1 Vehicle classification

The vehicle classification algorithm implemented using the "ssd mobilenet v1 coco" CNN as indicates in section 3.2.1, could be also used to classify with the generated scores for each of the four classes selected (see Fig:3.17).

As in the previous step, the classes of interest are only four, and the detections that correspond to other classes are eliminated.

- Class number 3 which corresponds to Car.
- Class number 4 which corresponds to Motorcycle.
- Class number 6 which corresponds to Bus.
- Class number 8 which corresponds to Truck.

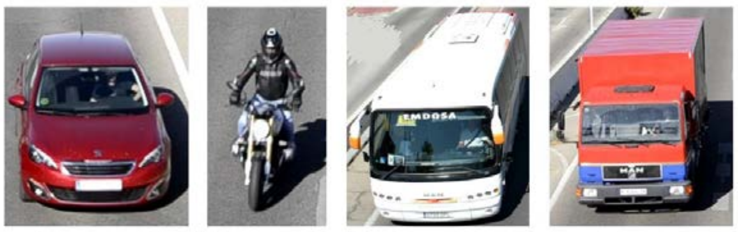


Figure 3.17: Example vehicles with the perspective of the front portico camera. The vehicles represented from left to right are motorcycle, car, truck and bus.

In this case, the classification is obtained directly in the detection stage, using the class score of its detection. The output class for each object detected corresponds to the class with the highest score.

3.4.2 Axle classification

The original vehicle detection CNN called "ssd mobilenet v1 coco" was retrained again on a wheel dataset (see Fig:3.7).

The retraining process is similar to the one described in Section:3.2.2, the CNN was retrained using Transfer Learning, keeping the first layers and only training the last layers.

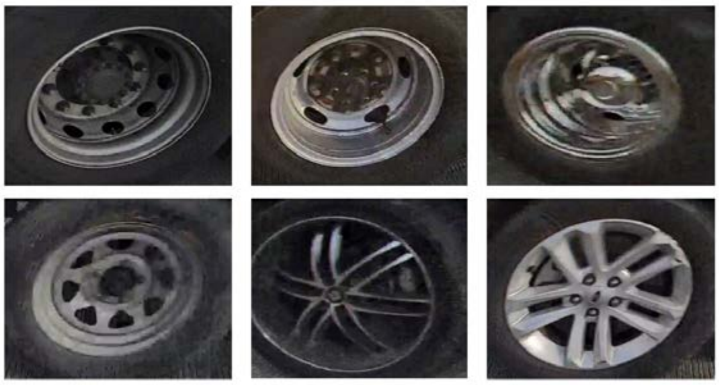


Figure 3.18: Subset of the axle classification images dataset. The first row corresponds to different double axles. The second row corresponds to different simple axles.

This retraining process is divided into different steps:

- Creating the dataset. The axle dataset was obtained (see Fig:3.7) by manually annotating the axles in all the video frames, like in Section:3.2.2. The difference with the previous dataset is that now the axles are labelled according to whether they are simple or double. This dataset is formed by 1.145 images, where 1.030 images correspond to simple axles and 115 correspond to double axles (see Fig:3.18).
- Training the model with the obtained dataset.
- Exporting the model so it can be run in the video processing algorithm codes. This model has two different output classes with their scores.

3.5 Plate recognition

3.5.1 Introduction

The used plate recognition software (for further references see www.arhungary.hu), has been bought by INDRA from the company ARH. Specifically, the algorithm purchased and used in this thesis is called 'CARMEN ANPR-Freeflow' which works with a neural network in order to obtain the license plates of a vehicle by using its image as an input.

This algorithm was selected for the implementation of this project since it provides many benefits, making it perfect for free-flow toll collection and traffic management systems. It was designed to recognize the plate license of a vehicle from any country in the world. It is able to detect the plate of vehicles that circulate at a speed of 250 km/h and, moreover, it works well in almost any scenario obtaining an accuracy of 98 percent in the evaluation carried out by that company.

First, this software searches the license plate in an input image and, then, it read its characters. The input image shall be a high-quality one, and the characters must be well appreciated. The processing time is not greater than a few hundredths of a second; hence, it is able to process multiple plates per second. Therefore, it is ideal to work in any kind of situation, even with a lot of traffic.

3.5.2 Optical Character Recognition (OCR) system development

For the same vehicle, T_*^k (T_F^k or T_B^k), the image used for the calculation of the OCR presents some differences with the one indicated by the bounding box of the tracker. These differences are due to the application of the following pre-processing steps:

1. The size of the image is augmented to help the car number plate detection.
2. The upper half of the image is eliminated assuming that the license won't be placed there. This step allows to avoid errors caused by reading wrong characters. A clear example of this problem are the driving school cars, which usually have character strings on the roof (see Fig:3.19). With this procedure we get that only the plate is detected as OCR.

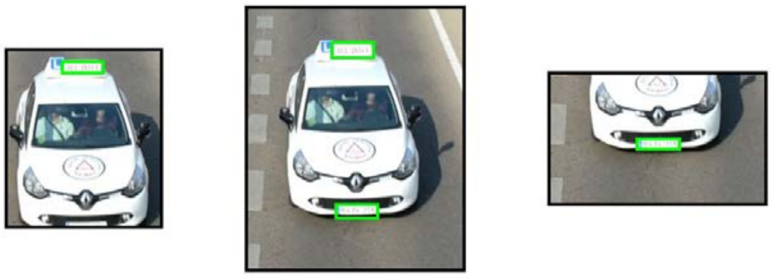


Figure 3.19: Example of image modification to calculate OCR. Starting from the left, in the first image, we have a plate miss detection, and, in the second one, we have a wrong detection in which it detects an another chain of characters as the plate.

To make the algorithm robust to errors, either caused by noise in the video capture step or by an OCR software error, the plate number is obtained several times. If in any case the registration is not detected, then, no result is saved. Thus, a new global system parameter H_{ocr} is considered which indicates how many times the OCR is calculated for each vehicle in each camera.

The final OCR output O_*^k is the combination of the different OCRs obtained. So, this algorithm is divided into two steps (see Fig:3.20):

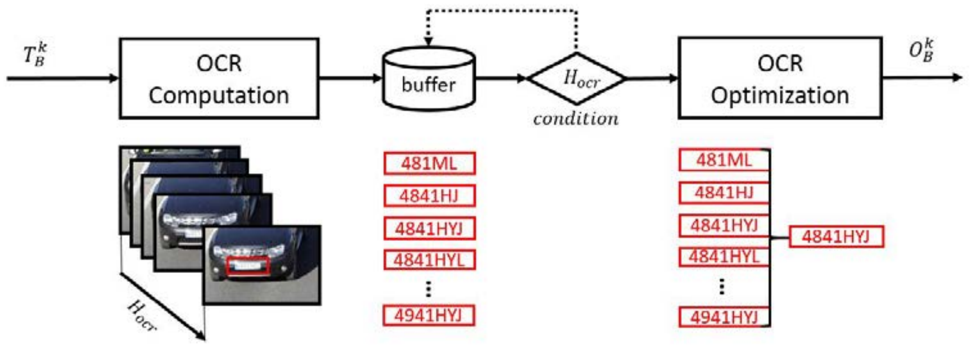


Figure 3.20: Scheme of the frontal device algorithm. When the buffer containing the plates of each vehicle is filled (has H_{ocr} plates), the optimization stage is performed to obtain a single output result.

1. OCR computation. The algorithm takes as input an image defined by the T_B^k or T_F^k (T_*^k) bounding boxes of a vehicle after the pre-processing steps, and obtains as output the OCR of the registration plate. These values are saved in the buffer for the H_{ocr} input images of the vehicle.

2. OCR optimization. Once the OCR has been calculated H_{ocr} times (the design condition is met, see Fig:3.20) for the same vehicle, the objective is to reduce the H_{ocr} number plates of this vehicle to only one. For this, the next steps are followed:

    - It is sought which of the others is the one that most resembles it. Each one of the registrations of the database it is compared with the rest, using a temporary filtering that finds the most common string given a series of noisy samples of that string. The initial plates of the buffer are changed by these new ones calculated.
    - The most repeated plate as the result of the previous operation is the final detection O_*^k.

3.6 Central System

This system is the one that carry out the unification of the three cameras, with the objective of having a unique information of ID, type, number/type of wheels, and the license number for each vehicle that crosses the portico.

The operation of the central system, requires the definition of two new elements, the database and the control signals for database updating called pistons. After, introducing both elements, we are going to analyze the updates that are made in the database by studying the algorithms of vehicle association and axle to vehicle association.

3.6.1 Piston

In a system of fixed cameras as the one posed, one can use a piston to interconnect and align camera processing. This piston is represented as a line in video processing algorithms (see Fig:3.21) and it is used to update the system database, helping to establish the class and the license plate assigned to each vehicle, as well as to compare times of the same vehicle along with the system of cameras.

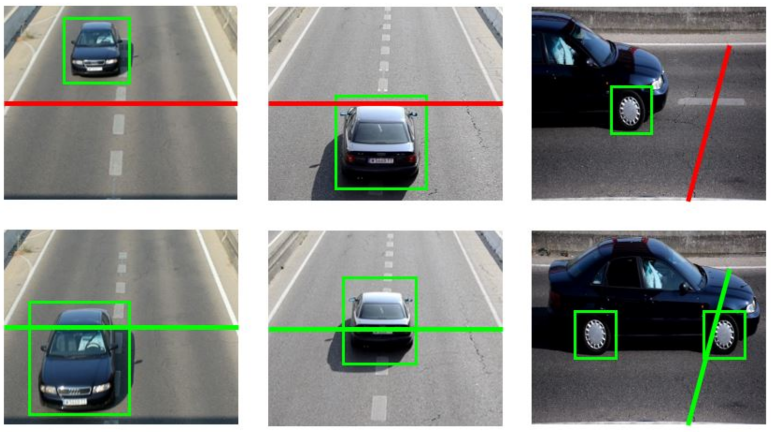


Figure 3.21: Example of pistons with different camera algorithms for the same vehicle. Starting from the left, the first column corresponds to the front camera, the second to the back camera and the third to the lateral camera. The upper row corresponds to the vehicle before passing through the piston and the lower row with the vehicle or their axles just at the moment they pass through the piston (green color of the piston).

The placement of the piston (see Fig:3.22) is in the place where the video is focused and close to the point of observation of the camera, the point of the image of the highest quality. The system is composed of three different pistons, one in each camera and its placement vary with each video processing algorithm.

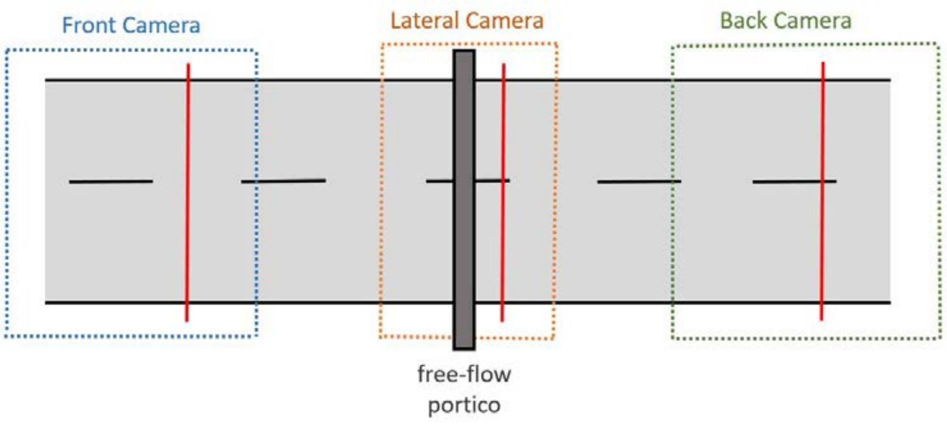


Figure 3.22: Example of colocation of the pistons (red lines) along with the camera system. the placement distance between the three pistons is similar.

The use of the piston in video processing algorithms is shown below. In each frame of the video, the class of each detected object (vehicle or axle) is calculated using one of the previous classification algorithms and assigned to a tracker in each time k. The class can vary a lot of times during its appearance on the screen (see Fig:3.23). To ensure that this class does not vary and there is only one class per object, the use of an auxiliary piston is employed.

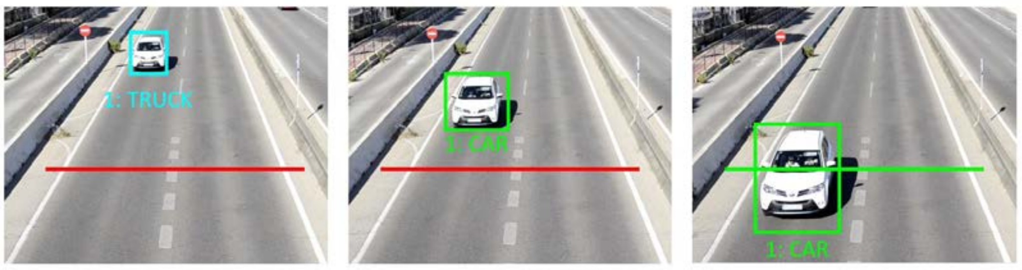


Figure 3.23: Example of vehicles crossing the piston line to established the class of each of them. The image on the right, represents when the object cross the piston line.

Once the object crosses the piston, the class that has been assigned a greater number of times to this tracker, is established as the class of the tracker that represent the object of interest (vehicle or axle) $[C]_{*}^{k}$, and the classification algorithm on that object is not run again.

In the front and back algorithms, the piston is also used to establish the registration of each vehicle. As seen in Section:, the OCR is calculated H_{ocr} times over each vehicle. Therefore, when a vehicle crosses the piston it gets its registration H_{ocr}

times, but only one enrollment result is obtained $[O]_F^k$, $[O]_B^k$ as explained in Section 3.6.1, with this you could avoid mistakes in obtaining it.

Apart from these reasons, the piston is essential to store the time in which each vehicle crosses each one of them, essential for the association algorithms.

Finally, all the results obtained (ID, class, plate and time of crossing by the piston of the vehicle) are established as the output $[V]_*^k$ that is sent to the central system.

Below are detailed schemes of the video processing algorithms studied above with the introduction of their respective pistons.

1. Frontal camera (see Fig:3.24). Obtain the data of each vehicle detected with the frontal camera $[V]_F^k$ (ID, class, plate number, and time in which it has passed through the piston P_F^k).

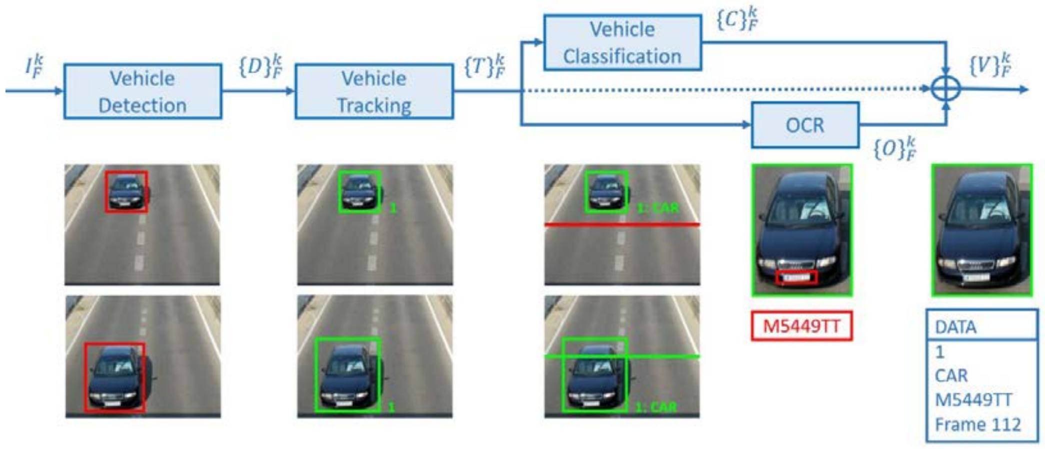


Figure 3.24: Scheme of the frontal video-processing algorithm.

2. Back camera (see Fig:3.25). Obtain the data of each vehicle detected with the back camera $[V]_B^k$ (ID, class, plate number, and time in which it has passed through the piston P_B^k).

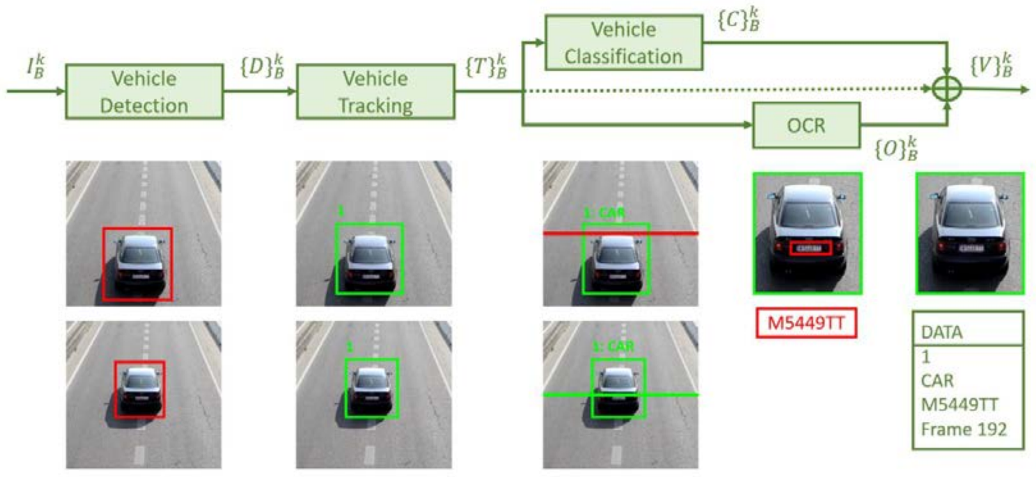


Figure 3.25: Scheme of the back video-processing algorithm.

3. Lateral camera (see Fig:3.26). Obtain the data of each axle detected with the lateral camera $[V]_L^k$ (ID, class, and time in which they passed through the piston P_L^k).

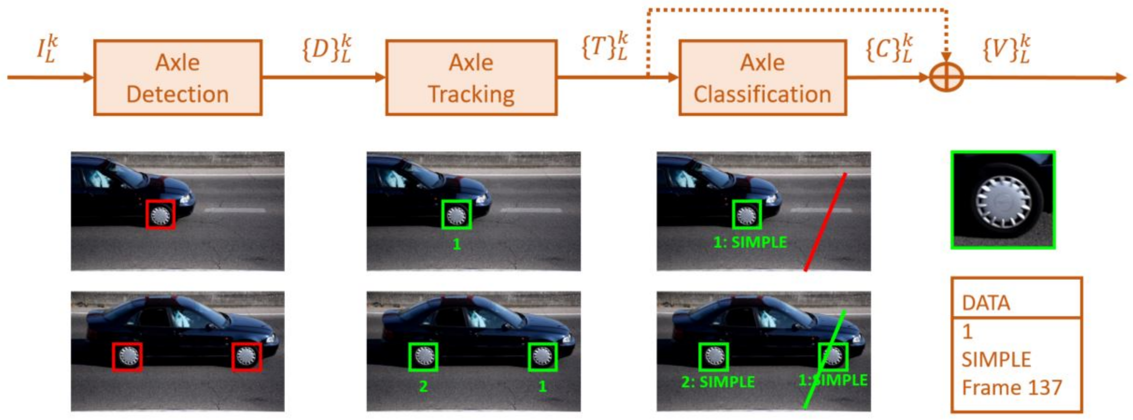
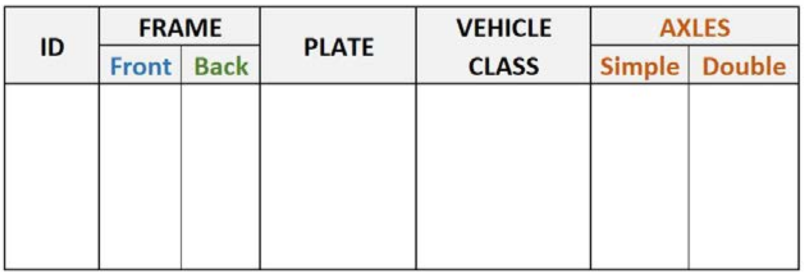


Figure 3.26: Scheme of the lateral video-processing algorithm.

3.6.2 Database

The central system is responsible for generating a database formed by objects of vehicle type. These objects have the next variables: ID, Time information (frame in which the object crosses the front and back pistons), number of plate, vehicle class and number of axles of each type for this vehicle (see Fig:3.27).



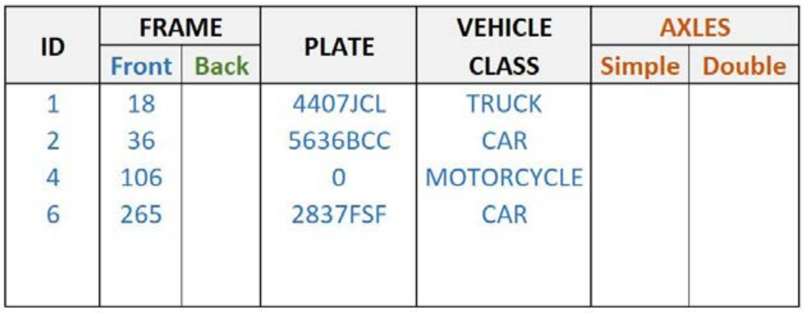
ID	FRAME		PLATE	VEHICLE CLASS	AXLES	
	Front	Back			Simple	Double

Figure 3.27: Example of the empty database.

The update of the database is done with the information provided by the video processing algorithms $[V]_*^k$, once the objects of interest cross the piston.

3.6.3 Vehicle Association

The central system receives vehicle information from the front $[V]_F^k$ and back $[V]_B^k$ cameras. Data from the front camera is used to generate a new vehicle in the database of the central system (see Fig:3.28).



ID	FRAME		PLATE	VEHICLE CLASS	AXLES	
	Front	Back			Simple	Double
1	18		4407JCL	TRUCK		
2	36		5636BCC	CAR		
4	106		0	MOTORCYCLE		
6	265		2837FSF	CAR		

Figure 3.28: Example of database refilled only with information from the front camera.

The association process consists of searching the database of a plate that matches the one provided by the back camera $[O]_B^k$.

In the event that a registration with these characteristics is found in the database, the information of the frame passing through the piston in the back camera is saved in the database (see Fig:3.29).

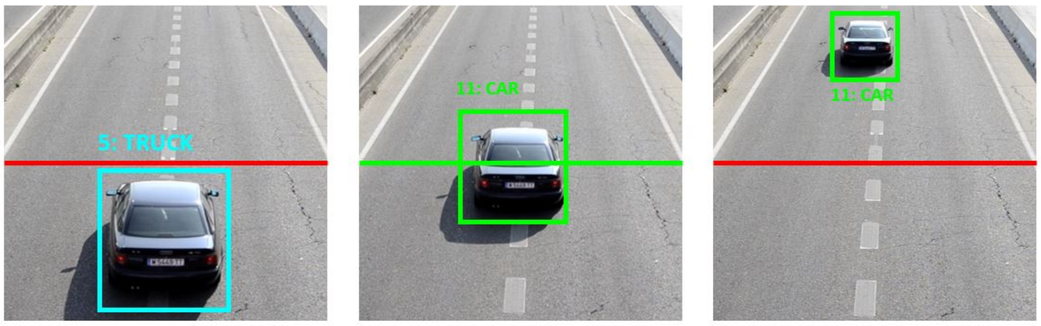
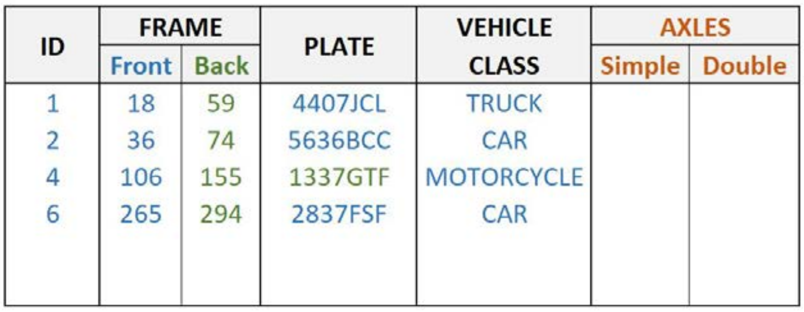


Figure 3.29: Example actualization data on the back camera algorithm due to the association vehicle front-back using the database. A change in the ID can be observed when crossing the piston due to the association process.

There is a case of special analysis, motorcycles. These do not have a front license plate, so the plate kept in the database by the front camera is '0'. To update the license plate correctly, if the back camera detects that the vehicle is a motorcycle, it is not searched by the plate in the database, it is searched by class. The last vehicle of the motorcycle class is searched in the database, obtaining the ID and modifying the plate to the one detected with the back camera (see Fig:3.30).



ID	FRAME		PLATE	VEHICLE CLASS	AXLES	
	Front	Back			Simple	Double
1	18	59	4407JCL	TRUCK		
2	36	74	5636BCC	CAR		
4	106	155	1337GTF	MOTORCYCLE		
6	265	294	2837FSF	CAR		

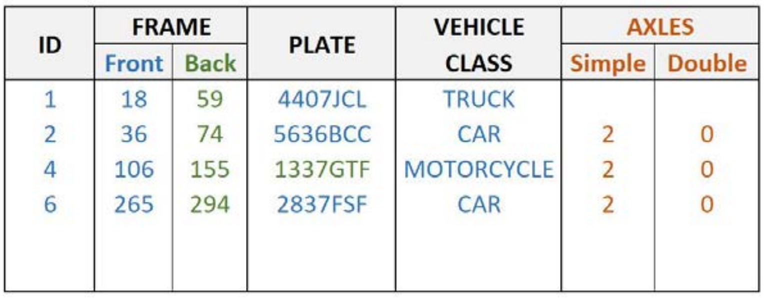
Figure 3.30: Example of the database Fig:3.28 after the vehicle association using the information of the back camera.

3.6.4 Axle to Vehicle Association

This algorithm performs the association of the vehicles with their axles, relying on the information of crossing time by the pistons P_*^k provided by the three algorithms of video processing.

To carry out this assignment two different methods are followed, depending on the class of the vehicle detected:

- Car and motorcycle. By definition, these vehicles always have two simple axes, cars with two wheels and motorcycles of one. So to the vehicles previously stored in the data with these characteristics are automatically assigned these values (see Fig:3.31). Completing the information of these vehicles $[V]^k_{F,B,L}$.



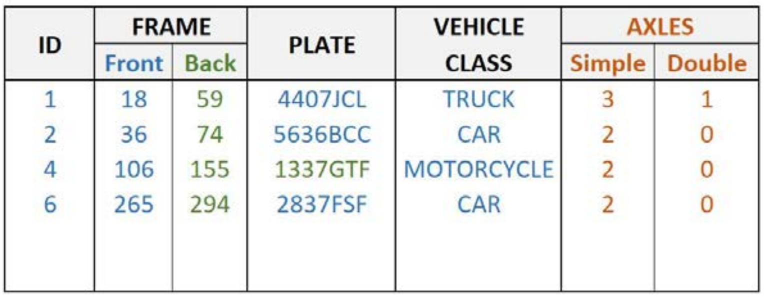
ID	FRAME		PLATE	VEHICLE CLASS	AXLES	
	Front	Back			Simple	Double
1	18	59	4407JCL	TRUCK		
2	36	74	5636BCC	CAR	2	0
4	106	155	1337GTF	MOTORCYCLE	2	0
6	265	294	2837FSF	CAR	2	0

Figure 3.31: Example of the database Fig:3.30 after the axle association for cars and motorcycles.

- Truck and bus. Using time information, we assign to the vehicle detected all the axles (differentiating between classes) that have been detected between the time that the vehicle has passed from the front camera to the back one (see Fig:3.32). That is if the detected axles meet the following equation are assigned to these vehicles (see Eq:3.1):

$$P^k_B > P^k_L > P^k_F \tag{3.1}$$

Where P^k_* represents the frames in the objects of interest cross the respective pistons. This time information P^k_* is contained within the database $[V]^k_{F,B}$ and in the information provided by the lateral camera $[V]^k_L$. To perform this assignment it is necessary that all three videos are temporarily adjusted. This completes the information of these vehicles in the database, obtaining $[V]^k_{F,B,L}$.



ID	FRAME		PLATE	VEHICLE CLASS	AXLES	
	Front	Back			Simple	Double
1	18	59	4407JCL	TRUCK	3	1
2	36	74	5636BCC	CAR	2	0
4	106	155	1337GTF	MOTORCYCLE	2	0
6	265	294	2837FSF	CAR	2	0

Figure 3.32: Example of the database Fig:3.31 after the axle association for trucks and buses.

Chapter 4

Development

4.1 Introduction

The hardware devices used to implement the portico are analyze in this chapter, as well as some of the most important features of the implemented software.

Free-flowing tracking infrastructures consist on a system with three KOMOTO C7S devices (see Fig:4.1) (see Section:4.3.2) used to perform the video-processing algorithms for each one of the cameras shown in Section 3.1. It is formed by:

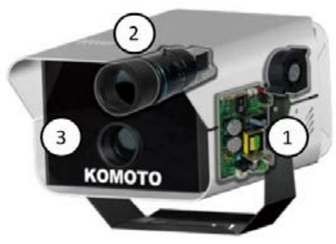


Figure 4.1: KOMOTO C7S with the different objects that make up this device (1: Jetson TX2, 2: Daheng Imaging Camera, 3: KOMOTO strobe).

- An embedded AI computing device Jetson TX2 (see . Section:4.2). It consist in some hardware provided with image processing software capable of detecting, classifying, identifying and tracking vehicles and axles on highways and in real-time. Also it contains the software necessary to extract the video input from the camera.
- A MERCURY USB3.0 series (MER-U3) camera (see Section:4.3.1) is DAHENG IMAGING's mature area scan industrial digital camera, featuring megapixels resolution, high definition, extremely low noise, perfect color conversion and

compact design.

- An integrated LED Strobe, which is used to illuminate the scene, allowing the operation during the night.

The central system runs in a KMDA-3602 computer (see Section:4.4). It is more powerful than Jetson TX2 boards, with better GPU to run CNNs. It is located in the portico.

Different Jetson TX2 boards are used in each portico to perform each video-processing algorithm, because of this hardware provides high speed working with CNNs when running them on the graphic processing unit (GPU) of the board, obtaining quick results although smaller than those obtained with a powerful computer. The low cost and the small size of this board compared with a traditional computer, make it a good choice for processing the frames of the different cameras of the system.

The algorithm designed on the JetsonTX2 board, is develop in Ubuntu with the programming languages python, C++ and Cython. The base of the algorithm is made in python and Cython is only used to include some functionalities programmed in C++.

Cython is a language used to generate CPython modules. The code is compiled in C or C++, and later via Cython interfaces, functions are generated that can be imported into python maintaining the same functionality and not involving a high computational cost.

Attending to the requests from the company in charge of this thesis, no further information about the software or the code developed to run the system will be included.

4.2 Jetson TX2

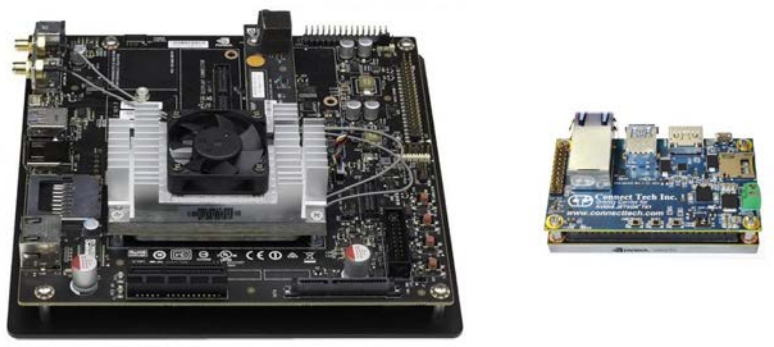


Figure 4.2: Both figures represents a device with a Jetson TX2 processor (the scale between them it is realistic). The device on the left corresponds to the developer kit, with all the possible benefits. The device on the right corresponds to a simpler board to integrate in the C7S KOMOTO device.

Jetson TX2 (for further references see www.nvidia.com) (see Fig:4.2) is the fastest and most powerful embedded AI device. It is possible to obtain real-time artificial intelligence (AI) performance thanks to this supercomputer with NVIDIA Pascal technology in one module. With a Denver 2/2 MB L2 dual-core HMP CPU, in addition to the quad-core ARM A57 / 2 MB L2 CPU, 4K x 2K 60Hz video encoding and decoding and 128-bit and 8GB LPDDR4 memory, Jetson TX2 is ideal for intelligent peripheral devices such as robots, drones, business collaboration, smart cameras and more.

- Size. It is possible to achieve exceptionally high calculation, accuracy and energy efficiency in a module the size of a credit card. Its small size of 50 mm x 87 mm allows real applications of deep learning in small format products, such as, for example, drones
- Performance. Due to the NVIDIA Pascal architecture of 256 cores and the memory of 8 GB of Jetson TX2, twice the performance and the energy efficiency of Jetson TX1 is achieve.
- Power. With Jetson TX2, it is possible to run large deep neural networks in order to achieve maximum accuracy in perimeter devices. With only 7.5 watts, it offers 25 times more energy efficient than a next-generation desktop CPU. This makes it the perfect choice for real-time processing in applications where bandwidth and latency can be a problem. These applications include factory

robots, commercial drones, business collaboration devices and smart cameras for smart cities.

4.2.1 Jetson TX2 SDK

The implementation of the design software has been carried out on both boards, first the installation was performed in the Developer Kit, and later on, in the board that was chosen to be a part of the C7S system (see Fig:4.2). This Jetson TX2 boards has its own software requirements, which is explained below (see Fig:4.3).

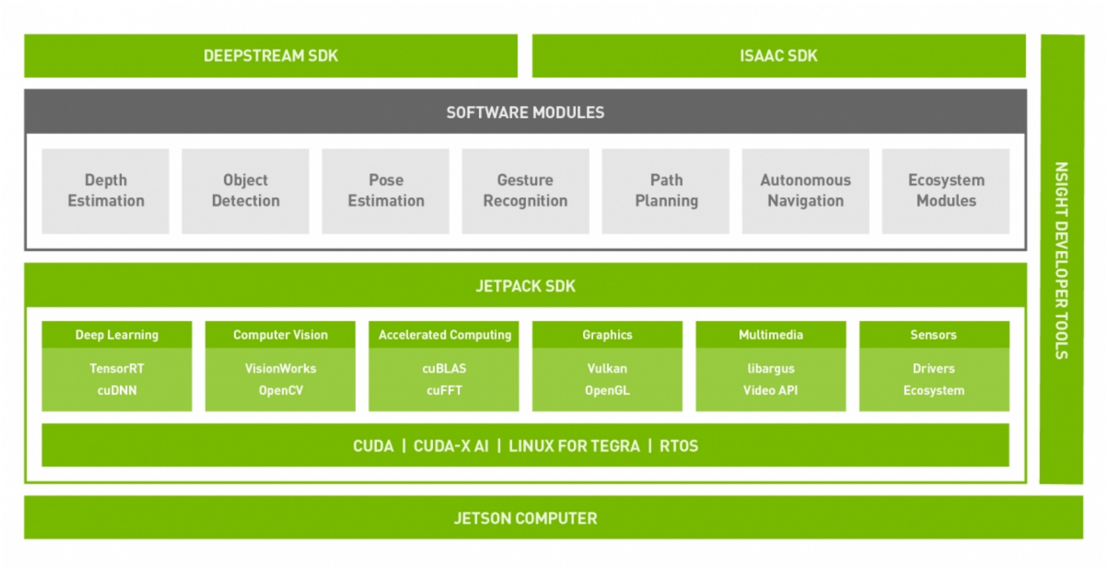


Figure 4.3: The Jetson Software Stack (for further references see www.developer.nvidia.com).

The Jetson platform software is based on the Jetpack SDK, which includes the Linux operating system, in our case Ubuntu 16.04, the board support package (BSP) and the NVIDIA CUDA. The used of DeepStream allows working efficiently with video analytics pipelines.

The best solution to build artificial intelligence (AI) applications on Jetson boards is the NVDIA Jetpack SDK. This software includes a lot of computer vision libraries like OpenCV, TensorRT, CUDA Toolkit, cuDNN or VisionWorks, making it easy to work with high-performance neural networks. In these boards has been installed Jetpack 3.3 version has been installed, which has the following important characteristics:

- OpenCV 4.0.
- TensorRT 4.0.

- CUDA 9.0.

To carry out the installation of the Jetpack 3.3, the steps followed are proposed in www.docs.nvidia.com, flashing the board with a computer host with the Ubuntu Linux x64 v16.04 operating system. All software must be compatible with the Jetson SDK.

The installation of TensorFlow 1.13.1 (with TensorRT support) has been done following the steps from www.docs.nvidia.com. This version has been chosen for its greater compatibility with the installed SDK.

For the implementation of the neural networks "ssd mobilenet v1 coco" introduced earlier (see Section:3.2.1) on this board, two models have been used, both programmed in python.

1.- TensorFlow

It remains the most popular deep learning framework today, based on the use of TensorFlow Object Detection API. This API is an open source framework built on top of TensorFlow in order to build, train and deploy object classification and detection models. The workflow that represents the used of TensorFlow on this algorithm is represented in (see Fig:4.4).

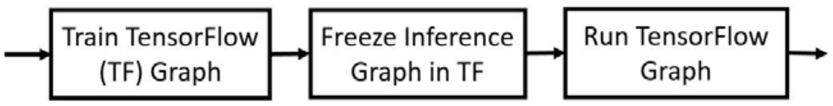


Figure 4.4: Inference workflow in TensorFlow.

First, the training of the network is done on TensorFlow, then the trained $frozen\_graph$ of the original network is generated. Finally, this graph is the one used to run CNNs on the software implemented in python.

2.- TensorRT

The change to the TensorRT model in Jetson TX2 devices was implemented to achieve an improvement in the performance of the CNNs. TensorRT speeds up TensorFlows deep learning inference through optimizations and high-performance runtimes for GPU-based platforms.

During the TensorFlow with TensorRT (TF-TRT) optimization, TensorRT performs different modifications to the original CNN graph:

- Layers whose result are not used are removed, to avoid unnecessary convolutional cost.

- Where possible, the convolutional, bias and ReLU layers shall be unified as a single layer whenever is possible.

The workflow that represents the used of TensorRT with TensorFlow is represented in (see Fig:4.5).

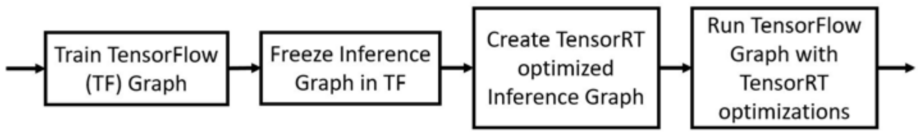

Train TensorFlow (TF) Graph
Freeze Inference Graph in TF
Create TensorRT optimized Inference Graph
Run TensorFlow Graph with TensorRT optimizations

Figure 4.5: Inference workflow in TensorRT.

The difference with the previous method, is that with this method once you get the graph of TensorFlow you introduce some simplifications and optimizations to transform it to TensorRT, generating the TensorRT $frozen\_graph$. This is the graph that is finally used to run CNNs.

The implementation of the TensorRT model can be done from the checkpoints and the config points of the original TensorFlow model, or from the $frozen\_graph$ created with this TensorFlow. Its implementation in python is based on the next algorithm www.github.com, and on the use of the function provided by the TensorRT library $trt.create\_inference\_graph()$ generating the final graph with $tf.import\_graph\_def()$.

The new integration use TensorRT from within TensorFlow, providing a simple and powerful API with different precision implementations FP32, FP16 or INT8. Use of half arithmetic precision or FP32, reduces the use of CNN memory compared to FP16 allowing the deployment of larger networks. As an example of this improvement in run time, we can look at the ResNet-50 network, where TensorRT with TensorFlow performs processing 8 times faster than only TensorFlow (see Fig:4.6).

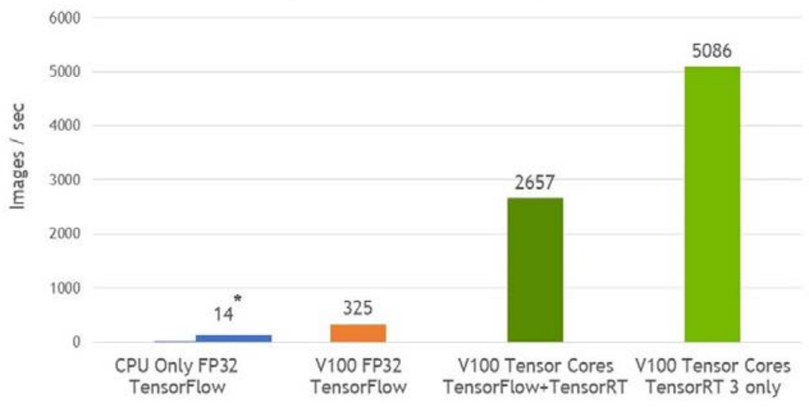


Figure 4.6: ResNet-50 performing comparative between the TensorFlow-TensorRT integration and running TensorFlow only (for further references see www.devblogs.nvidia.com).

Once the neural networks have been developed in the system, it is also necessary to talk about how OCR has been used and its implementation with Cython.

OCR software has been provided by Indra in the C++ language, which in turn obtained it from the company ARH. To make use of said software in the proposed design, a small program has been developed in C++ which receives an image as input and returns a char with the obtained number plate or '0' in case of not finding any.

This algorithm works with a key (pen-drive) provided by the manufacturer ARH, which must be connected to the hardware in order to obtain the OCR result of the registration.

To access this function of obtaining OCR created in C++ from the main program developed in python, Cython has been used. This language generates a function in python from the C++ function of obtaining of the plate, which can be called during the execution of the main program in python.

Cython has been used for obtaining the OCR created in C++ from the main program, developed in python. This language takes the OCR function in C++ and creates a similar one in python, which can be called during the execution of the main program.

4.3 Camera FLIR BFS-U3-31S4M-C

Figure 4.7: Camera FLIR BFS-U3-31S4M-C.

This model (for further references see www.flir.com) (see Fig:4.7) leverages Sony's Pregius global shutter CMOS technology. The Blackfly S combines the newest CMOS image sensors with the spinnaker software development kit. The characteristics of this camera can be summarized in:

- Flexibility and powerful auto-exposure controls with robust colour transformation tools thanks to the use of new on-camera image processing features and the latest CMOS sensors.
- Improvement of time per cycle compared to its predecessors using programmable logic and camera controls.
- Allow access to their GenICam3 API and GUI API library.

4.3.1 Camera Daheng Imaging MER-231-41U3M/C

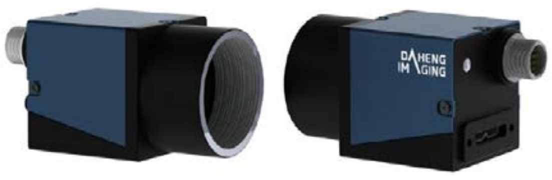


Figure 4.8: Camera Daheng Imaging MER-231-41U3M/C.

The MERCURY USB3.0 series(MER-U3) camera is DAHENG IMAGING's mature area scan industrial digital camera (for further references see www.daheng-imaging.com) (see Fig:4.8), featuring outstanding performance, compact design, extremely low noise and perfect color conversion.

The MER-231-41U3M/C camera is a monochrome/color USB3 Vision camera with the Sony IMX249 CMOS sensor. Thanks to the extremely compact (29mm x 29mm x 29mm), robust metal housings and locking screw connectors, the MERCURY cameras can secure the reliability of cameras deployed in harsh environments.

The MER-231-41U3M/C camera is powered over the USB interface. The MER-231-41U3M/C camera has opto-isolated I/Os. The GPIOs give MER-U3 maximum flexibility to adapt to specific needs. The camera has an outstanding price/performance ratio.

The MERCURY family cameras are especially suitable for machine vision applications such as industrial inspection, medical, scientific research, education, security and so on.

- Ultra small, light, robust with a very attractive price.
- 1/1.2" Global Shutter CMOS sensor.
- USB3.0 Interface, compatible with GenICam and USB3 Vision.
- 2.3 Megapixels
- 2 Programmable GPIOs

4.3.1.1 Camera Daheng Imaging SDK

This has been the camera chosen to be introduced into the C7S system, taking care of the video capture. The camera SDK provide by the manufacturers is compatible with the SDK provided by the Jetpack 3.3 to the board and with the Linux ArmV8 installed in it.

Using the functions provided by the SDK of the camera, a program was created in C++ able to obtain the images captured with the same. This program receives as an argument a file of the configuration of the camera where the film parameters can be chosen manually or automatically, as indicated in said file, and saves the images obtained in a chosen folder of the system.

The configuration file has the following parameters: exposure time, frame rate and gain. Apart from them, the focus and aperture parameters are adjusted directly from the lens of the camera.

- Exposure time or shutter speed. Is the period of time that the camera sensor is receiving light when we are taking the photo. This is important because depending on the ambient light we have to choose its value. In our case, we need short exposure times, which means that the photo is taken very quickly and the focused objects appear focused. It is necessary in the case of the axles, to avoid that these appear blurred due to their high speed.

- Frame rate. Images that capture the camera per second. Due to the high speed of the recorded objects, especially the axles, we need a high frame rate so that the tracker does not lose the detected objects.

- Gain, directly related to the ISO. Represents the sensitivity and modify it allows us to tell the sensor what sensitivity should have when collecting light. However, the higher the ISO, the easier it is for noise to appear in the photographs. In our case we do not have to alter it, the necessary luminosity that bring us the strobe is enough.

- Focus. The highest quality point of the recorded image, to focus is to make clear what is at a specific distance. In our case, it should be a wide area near the piston where the plates are read.

- Aperture. This parameter refers to the amount of light that aims to stop working until the sensor when taking a picture. The greater is, higher is the illumination of the filmed scene. Another aspect that serves the opening is to control the depth of field, which is the area of the image that appears focused where the larger the diaphragm aperture, the lower the depth of field.

Therefore, the correct selection of said parameters when recording the videos is crucial for the algorithms to work correctly. In (see Fig:4.9) we can observe different configurations of the camera parameters under analysis using the axles detection algorithm, observing how in some cases they are detected and in others not.

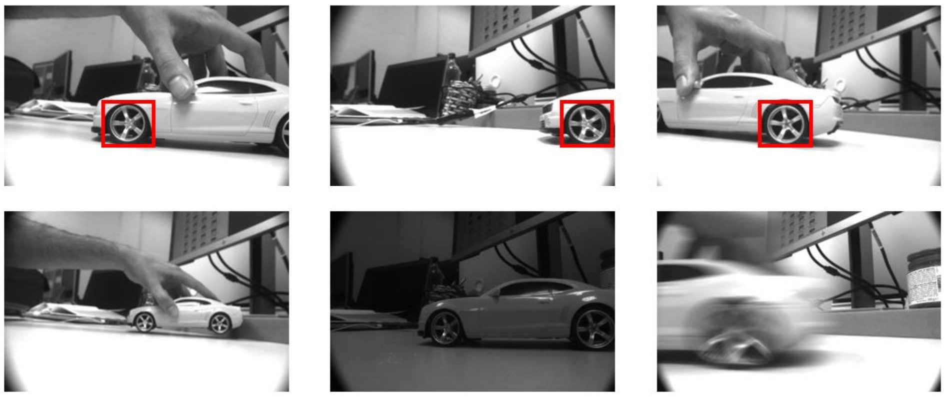


Figure 4.9: Example of toy car images captured with the Daheng Imaging Camera and with different parameters. Starting from the left, the first column corresponds to two images with the same focus and different vehicle positions, the second with two images with different aperture and the third with images with different time of exposure.

The camera is capable of capturing a maximum amount of 40 frames per second with good quality (see Fig:4.10). In this image we can see how the quality of the registration is high and therefore the possibility of obtaining a perfect OCR reading is also. Interest to have a reduced exposure time to catch vehicles in speed and a frame time as high as possible, depending on the processing speed of the image algorithms.

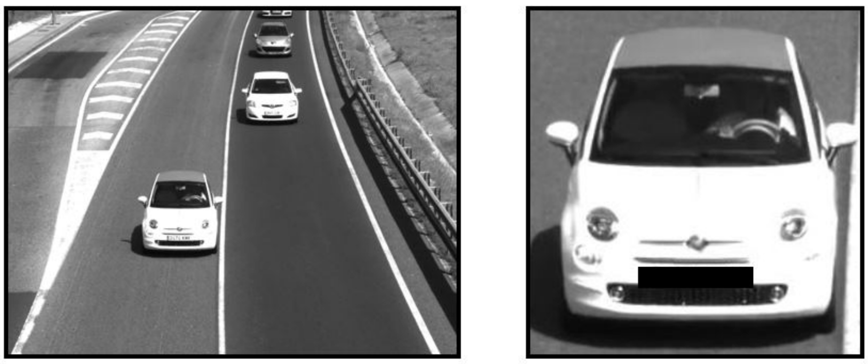


Figure 4.10: Example of image capture with Daheng Camera from a portico position. The image on the right is a zoom on the left image (the license plate has been blocked because it can not be shown)

4.3.2 KOMOTO C7S

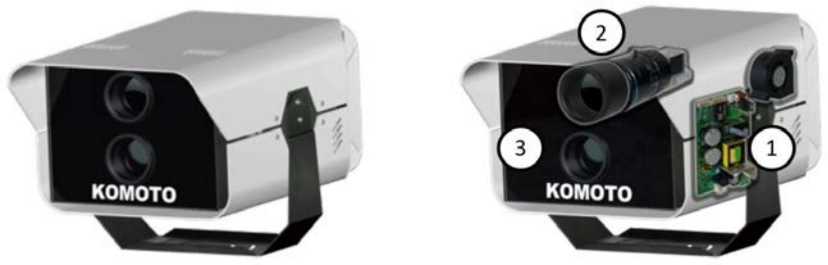


Figure 4.11: KOMOTO C7S. The left image represents the appearance and the right image represent the different objects that make up this device (1:Jetson TX2, 2:Daheng Imaging Camera, 3:KOMOTO strobe).

The building in Integrated Control Module (for further references see www.komoto.com) (see Fig:4.11), that holds the camera and the JetsonTX2 of the system, has the next specifications:

- Integrate Strobe with the latest High Power Led Technology. Light driver power up to 600W in pulse mode, with a wavelength of NIR 850nm.
- Trigger Mode (Flashes upon input pulse). 30us timing repetition rate for high frame rates. The trigger input could be from DC3.3V to 24V opto-isolated.
- Pulse width to 1.6ms at fully strobe.
- Overheat protection. The operating temperature supported is from -20 to +60 grades centigrade. Wide temperature detect range.
- Power requirement of DC24V and 2A.
- RS485 Interface.
- Support Device Holder and Weatherproof IP66.
- Dimensions (mm) of (L)239 x (W)352 x (H)232, and weight of 5kg.

4.4 KMDA-3602 Computer

Figure 4.12: KMDA-3602 Computer.

KMDA-3602 (for further references see www.armortec.net) (see Fig:4.12) is a high-performance In-vehicle computer from JHCTECH, with newest cooling design. It is powered by Skylake-S/Kabylake-S CPU, and Dual DDR4 2133MHz SODIMM memory.

It features NVIDIA or AMD MXM3.1 GPU module, 1*DP and 1*HDMI(Intel HD 4K), 3*DP and 1*HDMI (GPU 4K), 2*LAN, 4*POE,6*USB3.0, 2*Mini PCIe and 1*M.2 which support 4G,Wifi/BT and GPS communication, 1*mSATA and 1*2.5"SATA bay. DC6-48V wide power input, ITPS delay power on/off function. With CPU+GPU dual processor, multi LAN, multi-display, multi-wireless communication and anti-vibration design, it is suitable for mobile law enforcement vehicle and special engineering vehicle.

Between its characteristics highlight its GPU, MXM3.1 NVIDIA GTX/AMD RX series GPU module, support 3*DP1.2/1.3 and 1*HDMI 1.4b/2.0 4K/5K display. Which makes it perfect for working with convolutional neuronal networks.

4.5 Hardware system design

Once they have been studied the hardware systems and its software peculiarities that make up the system, it appears the global hardware design that conform the free-flow portico proposed in this thesis (see Fig:4.13).

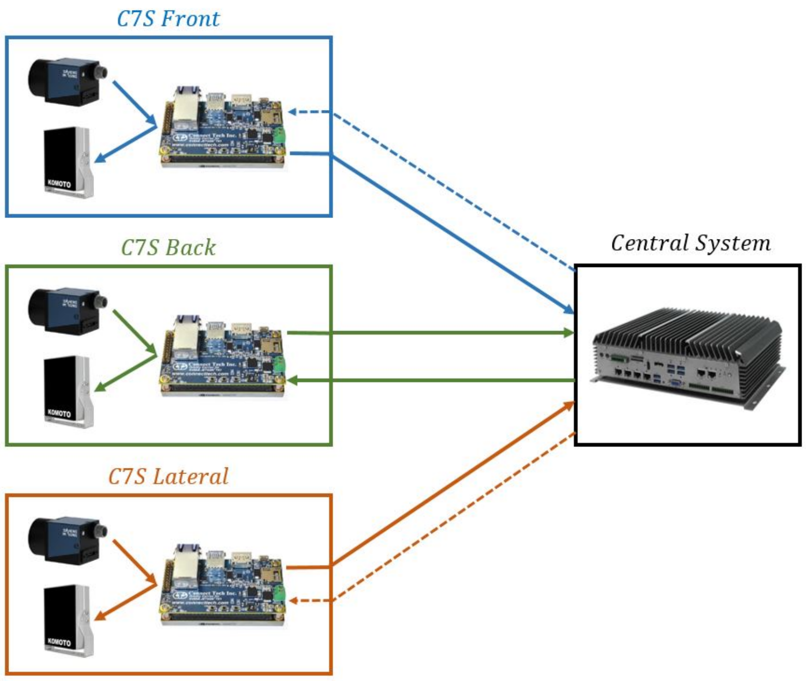


Figure 4.13: Scheme of the hardware connection system where each C7S correspond with one of the devices represented in Fig:4.11.

The communication that is made between each of the C7S is as follows. The Jetson TX2 gets the input images to be processed from the Daheng Imaging Camera, which are connected to the boards by using their USB 3.0 port. When there is little lighting the Jetson TX2 send a signal to the KOMOTO strobe to be lit.

The previous design shows two possible implementations of the system. One generated only by the continuous lines, which would correspond to a centralized software development in the central system, and another one that includes the striped lines that would compose a distributed software system. Therefore the central system is communicate with each of the cameras to receive information and send depending on the selected software design.

For the study of these proposals it is essential to know the meaning of a Data Transformation Services (DTS), which consists of a set of utilities and objects, which automate extract, transform and load operations to or from databases. This system can be equipped with all software capable of detect and classify, axles and vehicles from an input frame image. Initially, the system haves the CNNs explained above to detect and classify, based on the "ssd mobilenet v1 coco" network. The DTS would be located in the central system, receiving requests from the different Jetsons TX2 by a http petition and returning the results obtained.

These two proposals differ in the place where the neural networks of classification and detection developed in the three algorithms of the cameras are executed. Therefore the quality results obtained with both is the same, the difference between them is the computational cost.

4.5.1 Centralized System

The development of the detection and classification CNNs is developed in the central system, where the DTS is located. This DTS server is provided with the software to detect and classify, vehicle and axles.

The camera algorithms sent the input image to the DTS by a http petition, and the DTS returns the location of the interest objects in the image if there are. In addition to this information, the class information of said object also is returned, executing the classification CNNs (see Fig:4.14).

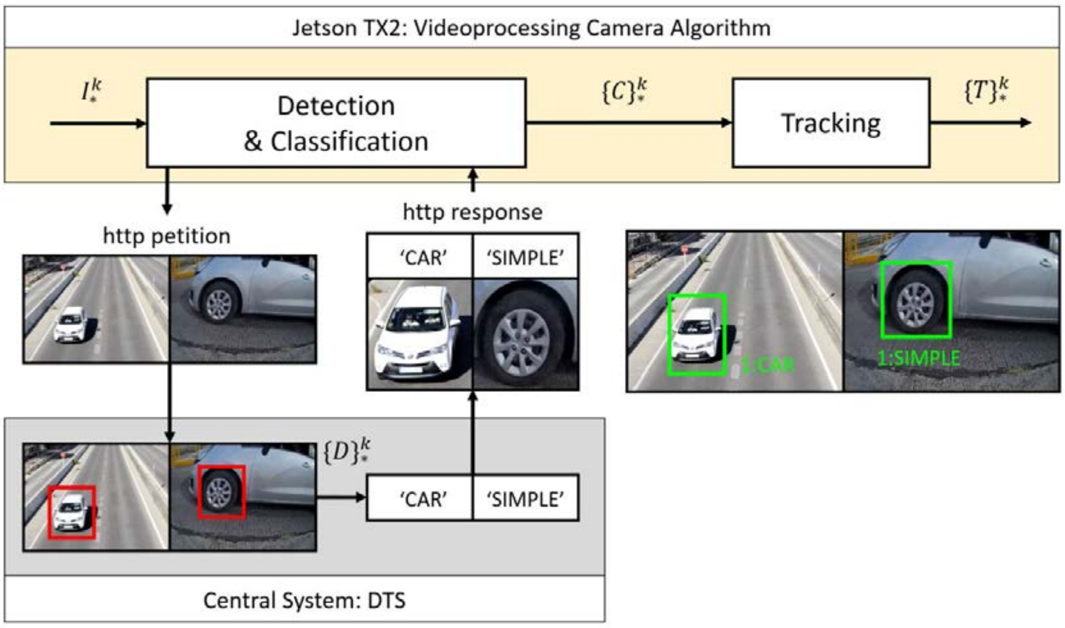


Figure 4.14: Example of how the DTS placed in the central system works for all the video-processing algorithms.

The central system has more power than the boards for the development of CNNs algorithms. But with a centralized system should be developing 6 CNNs in real time, in addition to making the communication between the different devices. This reason, along with the fact that getting implemented TensorRT in the Jetsons for the development of the CNN could get real-time processing, make the idea of making a distributed system more attractive.

In addition, the centralized system has the disadvantage that if the central system fails, the algorithm completely fails. While in a distributed system the image processing tasks are performed entirely in Jetsons, which could continue to obtain a portion of the results without the central system.

For those reasons, the system that has been chosen for the implementation is distributed.

4.5.2 Distributed System

In this system the central system do not have any DTS and therefore it do not take care of the tasks of detection and classification, only of the communication or between the different cameras to fill the database. This is the proposed system (see Fig:4.15).

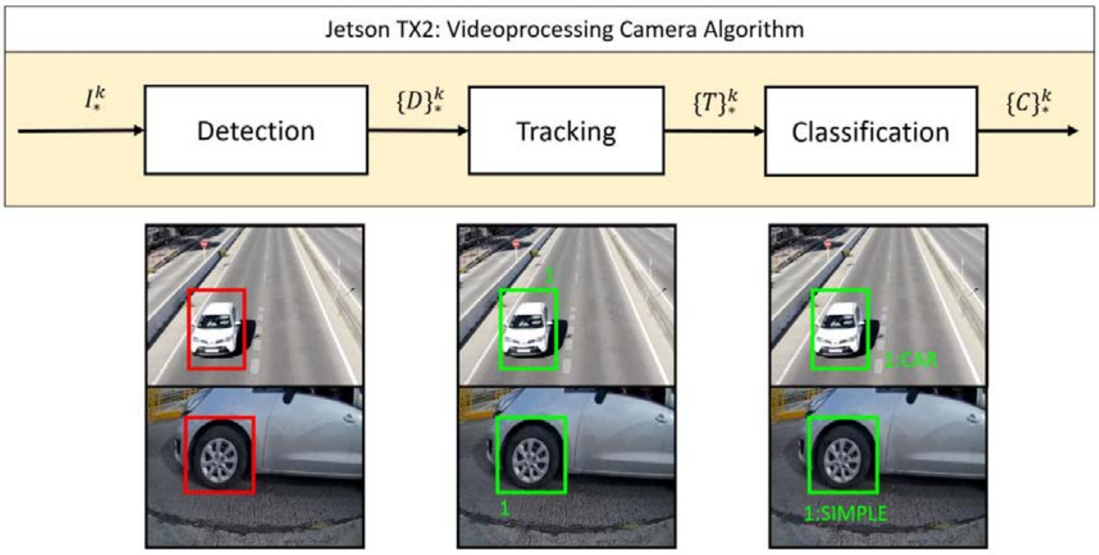


Figure 4.15: Example of how the algorithm works without DTS (proposed system see Section:3.1).

Chapter 5

Experimental evaluation

5.1 Introduction

First, an evaluation of the algorithms will be performed separately: detection and tracking, classification and reading of plates. Subsequently, an analysis of the results obtained for the association of vehicles and the association of them with their axles is also developed. The evaluation will be carried out on the data that save these algorithms in the database, i.e. their characteristics after crossing the piston.

It is an experimental evaluation based on video recorded with different cameras for two reasons:

1. The structure of the free-flow portico has not been designed in time, and therefore it has not been possible to have access to videos of the three Daheng Imaging cameras that would make up the system.

2. The lack of datasets focused on this structure, where the cameras are placed in the three positions fixed in the design and synchronized in time.

In the same way, the development of the entire algorithm is executed from a single plate Jetson TX2, the chosen one to introduce inside the structure C7S. Models of convolutional networks based on TensorRT have been generated, but these models have not been correctly introduced into the design. So the CNNs algorithms run over TensorFlow.

5.2 Setup framework

5.2.1 Datasets

The evaluation of the developed algorithms use three different datasets that aim to evaluate different parts of the system designed. The videos that form the dataset have been recorded with different cameras: Daheng Imaging, NIKKON D3500, NIKKON D5300 and CANNON 600D developing the function that would perform the cameras Daheng in the system of the original free-flow portico. The following sets of videos are proposed for evaluation:

- *Scenario*1 (see Fig:5.1). A single video recorded from the front position of the portico with a Daheng Imaging camera. With this video the objective is to check if the algorithm could work with the proposed camera. This has been the only video obtained so far with these cameras. This video of 7:22 minutes contains 238 vehicles, of which 225 are cars, 2 are buses, 4 are motorcycles and 7 are trucks.

- *Scenario*2 (see Fig:5.1). Consisting of two videos recorded simultaneously at 60 fps with NIKKON D3500 and NIKKON D5300 cameras, one located in the front position of portico and the other in the rear position. With this dataset the vehicles association and the back camera algorithms are evaluated. These videos of 4:39 minutes contains 89 vehicles, of which 87 are cars and 2 are trucks.

- *Scenario*3 (see Fig:5.1). Consisting of three videos that would simulate the placement of all the portico cameras. For the recording of this system the two previous cameras have been used at 60 fps in addition to a CANNON 600D, used to record the back at 25 fps. The recording characteristics of these cameras are different, without having the possibility of recording them with non-automatic capture parameters. The video of the back is of poor quality. The goal of this dataset is to show the overall performance of the algorithm. This videos of 10:00 minutes contains 163 vehicles, of which 158 are cars, 2 are buses, 2 are motorcycles and 1 is a truck.

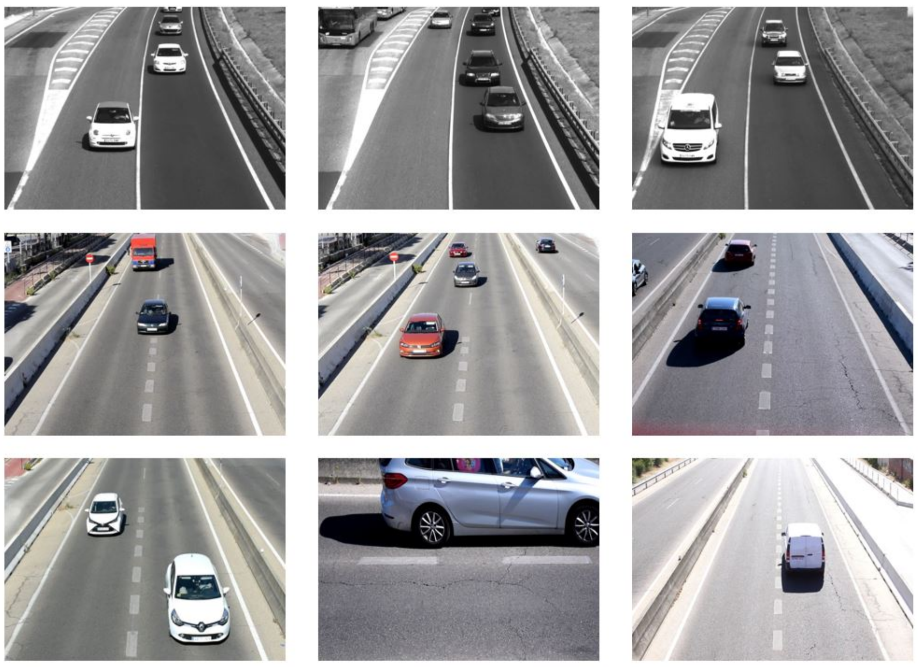


Figure 5.1: Subset of images from different $Scenarios$. Starting from above the first row corresponds to different images of the $Scenario1$, the second with $Scenario2$ and the third with the $Scenario3$.

In addition, ground truth (GT) files have been manually generated for each of these scenarios.

5.2.2 Metrics

The metrics used to evaluate classification, detection and tracking algorithms are recall and precision, its meaning can be seen in the following equations (see Eqs:5.1).

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

Where TP, TN, FP and FN means the number of true positives, true negatives, false positives and false negatives respectively.

To evaluate the percentage of success in reading plate license we measure the percentage of similarity between two character strings. This measure is obtained via python function $difflib\_get.close.matches()$ which have a parameter $cutoff$ that indicates the percentage of similarity between them. A percentage of 100% indicates that plates are identical, a percentage of 0% indicates that the strings do not have any common character.

5.2.3 Parameters review

As mentioned through the document, the following parameters need to be set for each scenario:

- TH_{score}. Threshold to remove detections that do not refer to objects of interest.
- TH_{iou}. Threshold to remove overlap between detections
- TH_{iou2}. Threshold to remove possible misassignments from trackers to detections.
- $minArea$. Parameter to remove detections with a lower area.
- H_{min}. Parameter to create trackers. If a tracker is assigned during H_{min} consecutive times to a detection, it is created.
- H_{max}. Parameter to remove trackers. If a tracker is not assigned to any detection during H_{max} consecutive times, it is removed.
- H_{ocr}. Number of times the OCR is calculated for each vehicle.
- P_1 and P_2. Points to determine the line that forms the piston. The piston is placed in the highest quality area of each of the videos, with the aim of obtaining the best results.

5.3 Performance evaluation of single camera Algorithms

5.3.1 Evaluation of front camera methods

1.- Experimental setup

For the evaluation of this algorithm the front video of $Scenario1$ has been chosen. The metrics used to study its operation are those introduced previously. The same parameters have been chosen for the two methods: $TH_{score} = 0.2$, $TH_{iou} = 0.3$, $TH_{iou2} = 0.1$, $minArea = 400$, $H_{min} = 3$, $H_{max} = 6$ and $H_{ocr} = 5$.

2.- Experimental results

The evaluation of the detection and tracking algorithm using the CNN method is shown in Table:5.1. The detection and tracking algorithm refers to the vehicle detections obtained after trackers cross the piston. Where $\#Vehicles$ is the number of vehicles that are in the video and $\#Detections$ is the number of them that are detected as it passes through the piston.

Vehicle detection and tracking			
#Vehicles	#Detections	Recall	Precision
238	226	0.9741	1

Table 5.1: Table with the results of detection and tracking algorithm with the CNN method for the video of the $Scenario1$.

The evaluation of the classification algorithm using the CNN method is shown in Table:5.2.

Vehicle classification				
Class	#Vehicles	#Detections	Recall	Precision
Car	225	204	0.9741	1
Bus	2	15	0.9375	0.5172
Motorcycle	4	0	0	0
Truck	7	6	0.5444	0.5

Table 5.2: Table with the results of classification algorithm with the CNN method for the video of the $Scenario1$.

The evaluation of the OCR method using the CNN method is shown in Table:5.3. Where $Distance(\%)$ indicates the similarity between the ground truth with the correct license plate value and the obtained license plate. $Distance = 100\%$ indicates that the chains are identical. $\#Hits$ indicates the number of plates correctly matched, $\#Misses$ indicates the number of plates not matched and $Success$ indicates the hit percentage.

Optical character recognition			
Distance (%)	#Hits	#Misses	Success (%)
100	201	25	88.94
90	210	16	92.92
80	217	9	96.02
70	221	5	97.79

Table 5.3: Table with the results of OCR algorithm with the CNN method for the video of the $Scenario1$.

The evaluation of the detection and tracking algorithm using the FGBG method is shown in Table:5.4.

Vehicle detection and tracking			
#Vehicles	#Detections	Recall	Precision
238	255	0.9586	0.8793

Table 5.4: Table with the results of detection and tracking algorithm with the FGBG method for the video of the *Scenario*1.

The evaluation of the classification algorithm using the FGBG method is shown in Table:5.5.

Vehicle classification				
Class	#Vehicles	#Detections	Recall	Precision
Car	225	210	0.8714	95.02
Bus	2	11	0.5	64.71
Motorcycle	4	0	0	0
Truck	7	11	0.61	52.38

Table 5.5: Table with the results of classification algorithms with the FGBG method for the video of the *Scenario*1.

The evaluation of the OCR method using the FGBG method is shown in Table:5.6.

Optical character recognition			
Distance (%)	#Hits	#Misses	Success (%)
100	221	34	86.66
90	232	23	90.98
80	241	14	94.51
70	247	8	96.86

Table 5.6: Table with the results of OCR algorithm with the FGBG method for the video of the *Scenario*1.

3.- Experimental discussion

Analyzing the results of detection and tracking, it is observed how the accuracy of the CNN algorithm is superior to that of FGBG. This is because while FGBG is an algorithm sensitive to changes in the scene, where any of them make bad vehicle detections, the CNN method is being used only to locate vehicles. The changes in lighting and focus that occur during the video generate many false positives in the FGBG method, making a bad performance (see Fig:5.2).

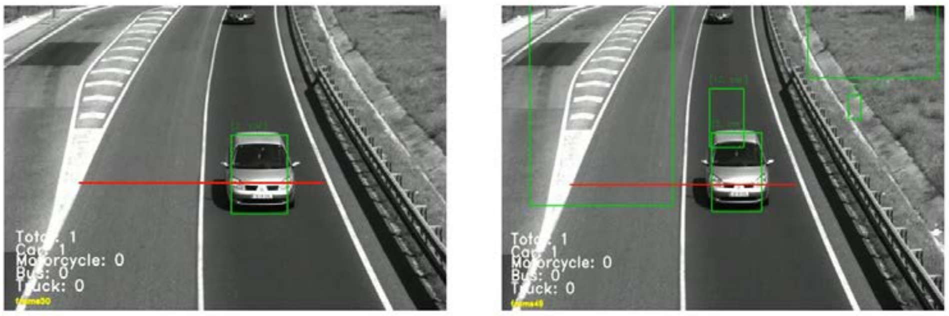


Figure 5.2: Example of results obtained by the algorithms of detection and traking in a change of illumination of the scene. The image on the left corresponds to the CCN method and the image on the left to the FGBG method.

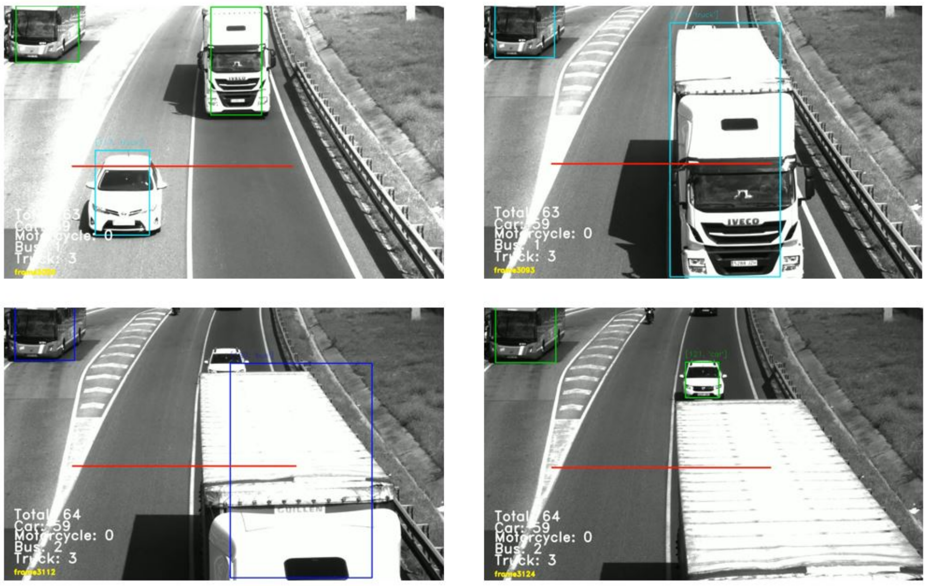


Figure 5.3: Example of the algorithm of detection and tracking with a truck during four frames of the video obtained of *Scenario*1. In the fourth frame the tracker is lost without correctly following the truck.

The failures produced in the detection with CNN, are due to occlusions or errors of the algorithm working with large detections (buses and trucks). Errors with buses and trucks, occurs because when only the tops of trucks and buses are visible, they are not detected and therefore the trackers are lost before it passes through the piston (see Fig:5.3). These errors also occur in the FGBG algorithm. To solve this problem a CNN retraining should be done with images of these parts of trucks and buses.

Both algorithms fail with motorcycles. The FGBG method fails because detections on motorcycles are eliminated due to their small size. The error of the CNN method is based on its training, since vehicles catalogued as motorcycles are mostly trained with images of lateral motorcycles, to solve this problem it would be necessary to re-train the CNN with new front and rear images of these vehicles (see Fig:5.4).

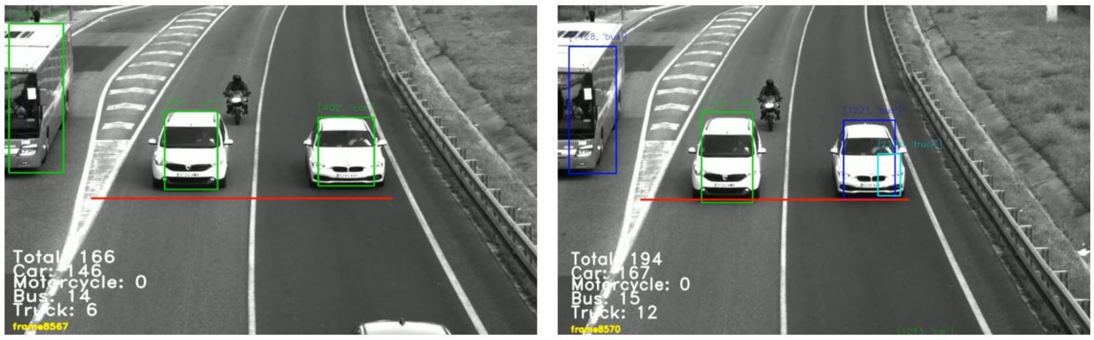


Figure 5.4: Example of failures of FGBG and CNN detection and tracking algorithms in the detection of motorcycles. The image on the left corresponds to CNN and the right to FGBG.

A large number of errors occur in the classification algorithm. This is due to the fact that the network used is not centered in the traffic area, since of the 90 classes that it detects only 4 are useful. There are a great number of mistakes in assigning bus and truck classes to cars. To solve this error, the ideal would be to make a training with a vehicle dataset that contains more vehicles with different positions.

The OCR algorithm works correctly because for 88.94% of vehicles in the case of CNN and for 86.66% of vehicles in the case of FGBG gets a perfect license plate reading, due to the high quality of the camera used for the video. Most are small errors in the estimation of some character. There are also four occlusion failures.

5.3.2 Evaluation of back camera methods

1.- Experimental setup

For the evaluation of this algorithm the rear video of $Scenario2$ has been chosen. The metrics used to study its operation are those introduced previously. Next the choice of the parameters is shown: $TH_{score} = 0.15$, $TH_{iou} = 0.3$, $TH_{iou2} = 0.1$, $minArea = 400$, $H_{min} = 3$, $H_{max} = 8$ and $H_{ocr} = 5$.

2.- Experimental results

The evaluation of the detection and tracking algorithm is shown in Table:5.7.

Vehicle detection and tracking			
#Vehicles	#Detections	Recall	Precision
89	85	0.9659	1

Table 5.7: Table with the results of detection and tracking algorithm for the back video of the *Scenario*2.

The evaluation of the classification algorithm is shown in Table:5.8.

Vehicle classification				
Class	#Vehicles	#Detections	Recall	Precision
Car	87	66	0.767	1
Bus	0	2	1	0.5
Motorcycle	0	0	0	0
Truck	2	17	1	0.5

Table 5.8: Table with the results of classification algorithm for the back video of the *Scenario*2.

The evaluation of the OCR method is shown in Table:5.9.

Optical character recognition			
Distance (%)	#Hits	#Misses	Success (%)
100	67	18	78.82
90	70	15	82.35
80	79	6	92.94
70	84	1	98.88

Table 5.9: Table with the results of OCR algorithm for the back video of the *Scenario*2.

3.- Experimental discussion

Analyzing the results of the detection and tracking algorithm it is observed that the algorithm works well for the detection of rear parts of cars, hitting 95.5% as it is shown in Table:5.7. It is important to note that the two trucks in the video have not been detected until they pass the piston (see Fig:5.5). To solve this problem I propose two options, to move the piston away from the portico, which would also cause worse results of obtaining OCR, or to perform a re-training of the CNN with images of these characteristics, as I have commented before.

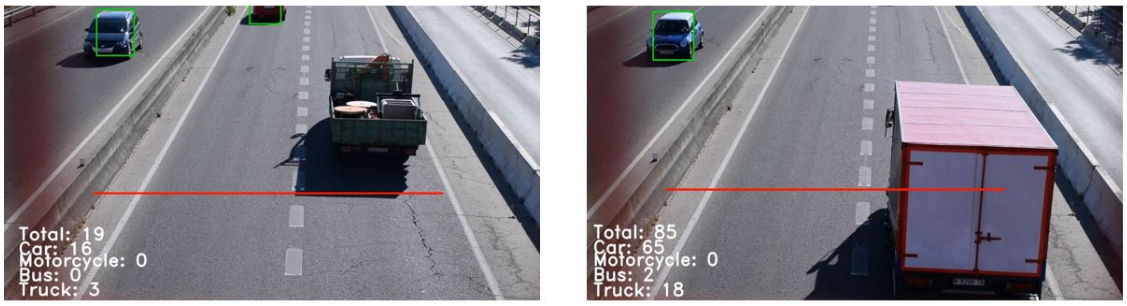


Figure 5.5: Example of two trucks not detected before passing through the piston.

As for the result of the classification, bad results are obtained. The problem is that cars are classified as trucks or buses a lot of times (see Fig:5.6), due to CNN's poverty in this regard. Re-training is proposed for improvement.

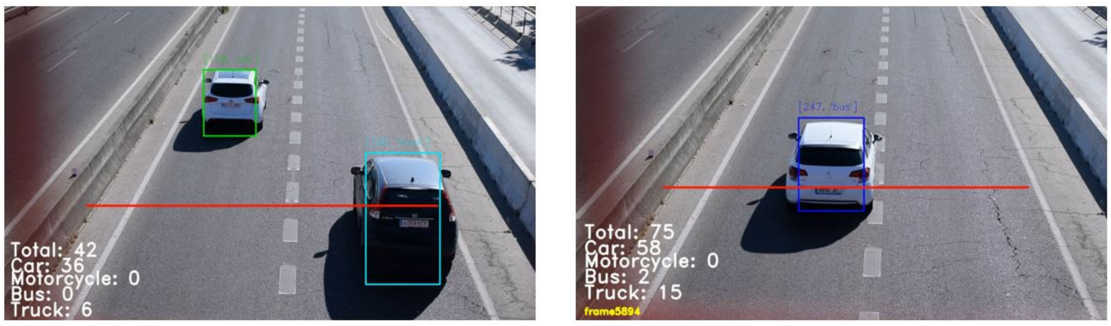


Figure 5.6: Example of two cars badly classified. The car in the image on the left is classified as a truck and the one in the image on the right as a bus.

The OCR algorithm works worse than in the previous case because only for 78.82% of vehicles are obtained the perfect plate, due to the lower quality of the camera used for the video.

5.3.3 Evaluation of lateral camera methods

1.- Experimental setup

For the evaluation of this algorithm the lateral video of *Scenario*3 has been chosen.

The metrics used to study its operation are those introduced previously. Next the choice of the parameters is shown: $TH_{score} = 0.1$, $TH_{iou} = 0.2$, $TH_{iou2} = 0.8$, $minArea = 100$, $H_{min} = 2$, $H_{max} = 3$ and $H_{ocr} = 5$.

2.- Experimental results

The evaluation of the detection and tracking algorithm is shown in Table:5.10.

Axle detection and tracking			
#Axles	#Detections	Recall	Precision
333	276	0.828	1

Table 5.10: Table with the results of detection and tracking algorithm for the lateral video of the $Scenario3$.

The evaluation of the classification algorithm is shown in Table:5.11).

Axle classification				
Class	#Axles	#Detections	Recall	Precision
Simple	329	273	0.9964	1
Double	4	3	1	0.75

Table 5.11: Table with the results of classification algorithm for the lateral video of the $Scenario3$.

3.- Experimental discussion

The negative part of this axle detection algorithm, is that it will always fail that there are occlusions of the same with other cars. However, in the video under study there are not axle occlusions; hence, missing axles are only due to detection errors. The algorithm performs with around 2% failure in axles detection, due to the reduced dataset used for creating the axle detection CNN, so some types of axles are never detected (see Fig:5.7) .

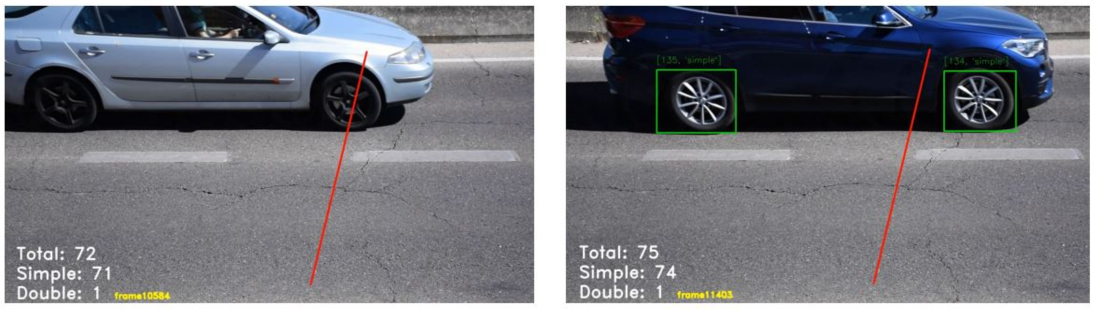


Figure 5.7: Example of the axle detection algorithm. The image on the left contains undetected axles and the image on the right contains detected axles.

As for the classification section, the results obtained are good, having only one mistake when classifying a double axle as simple. But also, it would be advisable to retrain the network with more training data (see Fig:5.8).

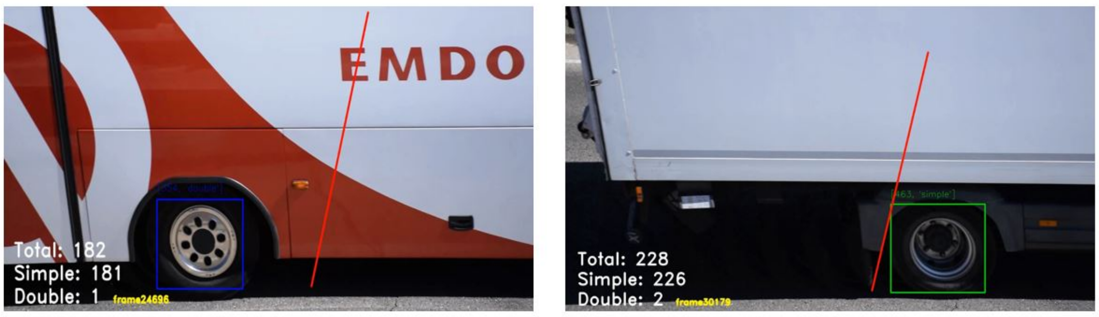


Figure 5.8: Example of the axle classification algorithm. The image on the left corresponds to a correct classification of a double axle and the image on the right corresponds to the error when classifying a double axle as simple.

5.4 Performance evaluation of vehicle association

1.- Experimental setup

The results shown, are obtained through the execution of the front and rear camera algorithms at the same time, and the use of the database introduced in previous sections. For the evaluation of this algorithm the videos of $Scenario2$ have been chosen. The metrics used to study its operation are those introduced previously. The same parameters have been chosen for back and front camera algorithms: $TH_{score} = 0.2$, $TH_{iou} = 0.3$, $TH_{iou2} = 0.1$, $minArea = 400$, $H_{min} = 3$, $H_{max} = 6$ and $H_{ocr} = 5$.

2.- Experimental results

The evaluation of the vehicle association is shown in Table:5.12. Where $\#ReID$ is the number of vehicles that have been correctly re-identified, $\#NOTReID$ the ones that have not been re-identified and $\#MisReID$ the failures that have occurred when re-identifying.

Vehicle association				
Distance (%)	# ReID	# NOT ReID	# Mis ReID	Success (%)
100	65	20	0	76.47
90	69	16	0	81.18
80	75	10	1	87.06
70	79	6	2	90.59

Table 5.12: Table with the results of vehicle association algorithm for the videos of the $Scenario2$.

The evaluation of the re-classification of the vehicles due to the algorithm of vehicle association using $Distance = 90\%$ is shown in Table:5.13.

Vehicle re-classification					
Class	#Vehicles	#Detections	#ReID	Recall	Precision
Car	87	79	69	0.9294	1
Bus	0	0	0	0	0
Motorcycle	0	0	0	0	0
Truck	2	6	0	1	0.5

Table 5.13: Table with the results of vehicle classification algorithm for the back video of the $Scenario2$ after the re-identification process.

3.- Experimental discussion

A re-identification algorithm based on the license plate must have a very high percentage of $Distance$, to not make mistakes when re-identifying. As you can see in Table:5.12, the perfect re-identification ($Distance = 100\%$) of license plate get a 76% success rate, due to errors in the reading of the license plate. These errors could be solved using the recording cameras with the appropriate configuration.

Using a percentage lower than 90% will never be a possibility of implantation in the algorithm. In this particular case, the percentage of 90% has been chosen to perform the algorithm because no failures have been made $\#MisReID = 0$.

Table:5.12 shows the classification of the rear camera after the re-identification process. To measure its performance, a comparison is made with the results obtained for the same video with the algorithm of the back camera previously analyzed (see Table:5.8). Through this process, 14 classifications have been corrected, changing 12 classes from truck and 2 classes from bus, to car (see Fig:5.9).

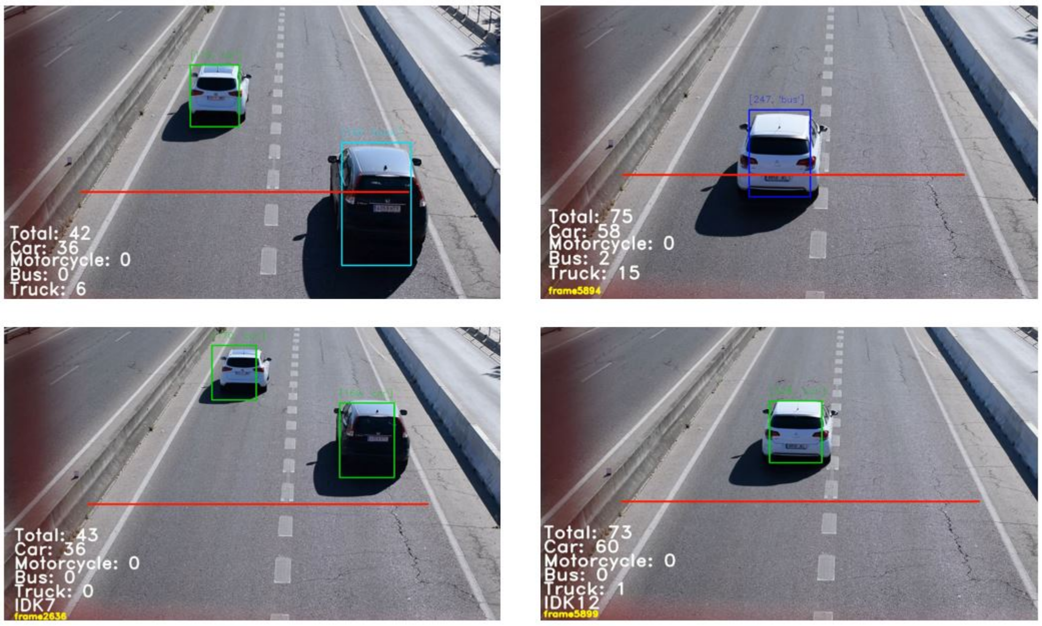


Figure 5.9: Example of correct re-identification. The two upper images correspond with the result of the single back algorithm and the two lower images with the result after the re-identification process.

5.5 Performance evaluation of axle to vehicle association

1.- Experimental setup

The results shown, are obtained through the execution of the front, back and lateral cameras at the same time and the use of the database introduced in previous sections. For the evaluation of this algorithm the videos of $Scenario2$ and $Scenario3$ has been chosen. The metrics used to study its operation are those introduced previously. The same parameters have been chosen for back and front camera algorithms: $TH_{score} = 0.2$, $TH_{iou} = 0.3$, $TH_{iou2} = 0.1$, $minArea = 400$, $H_{min} = 3$, $H_{max} = 6$ and $H_{ocr} = 5$. The following parameters are used for the axles algorithm: $TH_{score} = 0.1$, $TH_{iou} = 0.2$, $TH_{iou2} = 0.8$, $minArea = 100$, $H_{min} = 2$, $H_{max} = 3$ and $H_{ocr} = 5$.

2.- Experimental results

The evaluation of the axle to vehicle association in the $Scenario3$ is shown in Table:5.14. Where $\#A.Simples$ is the number of simple axles detected and $\#A.Double$ is the number of double axles detected. Success (%) is the the percentage of success in the axles assignment to the correspond detected vehicles.

Axle to vehicle association					
Class	# Vehicles	# ReID	# A.Simples	# A.Doubles	Success (%)
Car	158	15	30	0	100
Bus	1	0	0	0	0
Motorcycle	2	0	0	0	0
Truck	2	0	0	0	0

Table 5.14: Table with the results of axle to vehicle association algorithm for the videos of the $Scenario3$.

The evaluation of the axle to vehicle association in the $Scenario2$ is shown in Table:5.15.

Axle to vehicle association					
Class	# Vehicles	# ReID	# A.Simples	# A.Doubles	Success (%)
Car	87	69	138	0	100
Bus	0	0	0	0	0
Motorcycle	0	0	0	0	0
Truck	2	0	0	0	0

Table 5.15: Table with the results of axle to vehicle association algorithm for the videos of the $Scenario2$.

3.- Experimental discussion

Two processes are used to assign axles to vehicles: in one hand two simple axles are assigned to each car and re-identified motorbike, on the other hand, trucks, and buses are assigned axles that have been detected during the time it takes for the vehicle to move from the front camera piston to the rear camera piston.

For the allocation of axes to cars and motorcycles, 100% accuracy will always be achieved, however, in the case of trucks and buses, there may be errors. These errors will generally be due to the capture of axles of other vehicles, during the time in which the truck or bus is between the front and rear camera.

In this case, the axles have been assigned to the re-identified cars. However, it has not been possible to carry out the complete system because there are no examples recorded with three good quality cameras containing trucks or buses.

The *Scenario*3 is made up of three videos recorded synchronously, but the quality of the rear video is poor due to the illumination and the automatic focus of the camera used. It only detects 42 vehicles of the 163 that contains the video, and only 15 of them are re-identified (see Fig:5.10). The *Scenario*2 has no side camera video.

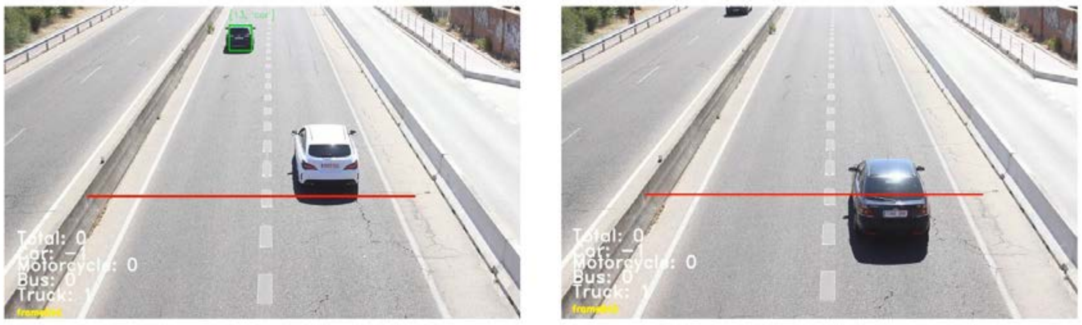


Figure 5.10: Example of two undetected vehicles with the back camera algorithm for the *Scenario*3 rear video.

Chapter 6

Conclusions and future work

6.1 Conclusions

This project describes a system to control traffic using a multi-camera system. The scope of application is a free-flow portico scenario. The system integrates software solutions for the detection, classification, tracking and re-identification of vehicles in highways. These methods are integrated on a hardware setup using Jetson TX2 boards and Daheng Imaging cameras. Finally, a set of initial strategies to combine information among the cameras are proposed.

The project starts by reviewing state-of-the art solutions on the involved techniques and follows by describing the processing pipeline and the hardware technology. Finally, a simulated experimental evaluation of the individual methods and of the combination strategies is carried out.

Experimental results indicate that the processing pipeline of each camera achieves decent performance and is able to handle capture challenges as illumination changes and occlusions. However, the deep learning solutions used along the project should be retrained before their use on the real scenario. On the other hand, the combination schemes performed on the central system are highly dependent of the individual pipelines. Their current version lacks of the robustness to be used on the free-flow portico scenario.

6.2 Future work

Software multi-camera design implementation still has a long way to go. The first step will be to obtain real videos, properly recorded with the cameras Daheng Imaging, with which to evaluate the algorithm correctly.

It will be necessary to improve the performance of all the neuronal networks used, both to detect and to classify. For its improvement is proposed a retrain of these networks, creating a large dataset with images of all categories of vehicles and axles, chosen from the different portico positions from which these objects will be visible. In addition, these images should be taken at different times of the day, with the aim of obtaining a robust CNN to lighting or weather changes.

It proposes an improvement, studying new procedures of re-identification for the association of vehicles, and multi-camera systems for the association of axles to vehicles.

It will be necessary for real-time processing. For this purpose, among other things, the TensorRT models of the CNNs used must be correctly implemented, in order to exploit the characteristics of the Jetson TX2 boards.

We will continue in the development of this algorithm with the objective of being implemented in real porticoes in which INDRA is working.

Bibliography

[1] Z. Tang, M. Naphade, M. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. C. Anastasiu, and J. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," *CoRR*, vol. abs/1903.09254, 2019.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov 2004.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, pp. 346–359, June 2008.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, (Washington, DC, USA), pp. 886–893, IEEE Computer Society, 2005.

[5] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017. PMID: 28599112.

[6] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2* (D. S. Touretzky, ed.), pp. 396–404, Morgan-Kaufmann, 1990.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[8] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, "Overfeat: Integrated recognition, localization and detection using convolutional net-

works," in *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[9] J. Zhang, F. Wang, K. Wang, W. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 1624–1639, 12 2011.

[10] H. Van Pham and B.-R. Lee, "Front-view car detection and counting with occlusion in dense traffic flow," *International Journal of Control, Automation and Systems*, vol. 13, pp. 1150–1160, Oct 2015.

[11] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "CNN-RNN: A unified framework for multi-label image classification," *CoRR*, vol. abs/1604.04573, 2016.

[12] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," *CoRR*, vol. abs/1504.02340, 2015.

[13] M. Valera and S. Velastin, "Intelligent distributed surveillance systems: A review," *Vision, Image and Signal Processing, IEE Proceedings -*, vol. 152, pp. 192 – 204, 05 2005.

[14] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, pp. 38:1–38:55, Sept. 2014.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[16] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," *CoRR*, vol. abs/1312.2249, 2013.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[18] X. Hu, X. Xu, Y. Xiao, H. Chen, S. He, J. Qin, and P. Heng, "Sinet: A scale-insensitive convolutional neural network for fast vehicle detection," *CoRR*, vol. abs/1804.00433, 2018.

[19] B. Li, T. Wu, and S.-C. Zhu, "Integrating context and occlusion for car detection by hierarchical and-or model," in *Computer Vision – ECCV 2014* (D. Fleet,

T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 652–667, Springer International Publishing, 2014.

[20] M. Vargas, J. M. Milla, S. L. Toral, and F. Barrero, "An enhanced background estimation algorithm for vehicle detection in urban traffic scenes," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 3694–3709, Oct 2010.

[21] A. Geiger, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, (Washington, DC, USA), pp. 3354–3361, IEEE Computer Society, 2012.

[22] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[23] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," *CoRR*, vol. abs/1607.07155, 2016.

[24] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[25] M. Shaker and M. ElHelw, "Optical character recognition using deep recurrent attention model," in *Proceedings of the 2Nd International Conference on Robotics, Control and Automation*, ICRCA '17, (New York, NY, USA), pp. 56–59, ACM, 2017.

[26] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," *arXiv e-prints*, p. arXiv:1512.02325, Dec 2015.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Confer-*

ence on Neural Information Processing Systems - Volume 1, NIPS'12, (USA), pp. 1097–1105, Curran Associates Inc., 2012.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[31] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, pp. 843–854, December 1979.

[32] Y. Bar-Shalom, *Tracking and Data Association.* San Diego, CA, USA: Academic Press Professional, Inc., 1987.

[33] S. Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid, "Joint probabilistic data association revisited," in *2015 IEEE International Conference on Computer Vision (ICCV)*, (Los Alamitos, CA, USA), pp. 3047–3055, IEEE Computer Society, dec 2015.

[34] F. Heymann, J. Hoth, P. Banys, and G. Siegert, "Validation of radar image tracking algorithms with simulated data," *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, vol. 11, no. 3, pp. 511–518, 2017.

[35] M. A. Naiel, M. O. Ahmad, M. Swamy, J. Lim, and M.-H. Yang, "Online multi-object tracking via robust collaborative model and sample selection," *Comput. Vis. Image Underst.*, vol. 154, pp. 94–107, Jan. 2017.

[36] H. W. Kuhn, "The hungarian method for the assignment problem," in *50 Years of Integer Programming*, 2010.

[37] B. Sahbani and W. Adiprawita, "Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system," 10 2016.

[38] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[39] X. Liu, W. Liu, H. Ma, and H. Fu, "Large-scale vehicle re-identification in urban surveillance videos," *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2016.

[40] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (alpr): A state-of-the-art review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, pp. 311–325, Feb 2013.

[41] S. M. Silva and C. R. Jung, "License plate detection and recognition in unconstrained scenarios," in *2018 European Conference on Computer Vision (ECCV)*, Sep 2018.

[42] J. Bromley, J. Bentz, L. Bottou, I. Guyon, Y. Lecun, C. Moore, E. Sackinger, and R. Shah, "Signature verification using a siamese time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, 8 1993.

[43] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546 vol. 1, June 2005.

[44] C. Zhang, W. Liu, H. Ma, and H. Fu, "Siamese neural network based gait recognition for human identification," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2832–2836, 2016.

[45] O. Javed, K. Shafique, Z. Rasheed, and M. Shah, "Modeling inter-camera space-time and appearance relationships for tracking across non-overlapping views," *Comput. Vis. Image Underst.*, vol. 109, pp. 146–162, Feb. 2008.

[46] G. Kayumbi, P. L. Mazzeo, P. Spagnolo, M. Taj, and A. Cavallaro, "Distributed visual sensing for virtual top-view trajectory generation in football videos," in *Proceedings of the 2008 International Conference on Content-based Image and Video Retrieval*, CIVR '08, (New York, NY, USA), pp. 535–542, ACM, 2008.

[47] R. Eshel and Y. Moses, "Tracking in a dense crowd using multiple cameras," *International Journal of Computer Vision*, vol. 88, pp. 129–143, May 2010.

[48] P. Peng, Y. Tian, Y. Wang, J. Li, and T. Huang, "Robust multiple cameras pedestrian detection with multi-view bayesian network," *Pattern Recogn.*, vol. 48, pp. 1760–1772, May 2015.

[49] A. Lopez-Cifuentes, M. Escudero-Viñolo, J. Bescós, and P. Carballeira, "Semantic driven multi-camera pedestrian detection," *CoRR*, vol. abs/1812.10779, 2018.

[50] "El tráfico en las autopistas de peaje," *Dirección general de carreteras, Ministerio de fomento, Gobierno de España*, 2010.

[51] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," vol. 2, pp. 28 – 31 Vol.2, 09 2004.

[52] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recogn. Lett.*, vol. 27, pp. 773–780, May 2006.