

**École polytechnique de Louvain**

# **Multi-Camera Parking lot Vehicle Re-Identification with a siamese network**

Author: **Benoît GÉRIN**

Supervisor: **Benoît MACQ**

Readers: **Dani MANJAH, Christophe DE VLEESCHOUWER,  
Christophe PARISOT, Jonathan SAMELSON**

Academic year 2020–2021

Master [120] in Electrical Engineering

## **Abstract**

This thesis aims to propose a vehicle re-identification algorithm that is using the information of several cameras placed around a parking lot to track multiple vehicles over time. To this end, a siamese deep neural network is designed and trained to model the similarity between two images. In addition, several metrics are proposed to quantify the quality of the information contained in a multi-view vehicle representation. Furthermore, a parking lot dataset is recorded to test the algorithm in real conditions. The experiments show on one hand the improvement brought by the number of cameras and the multi-view vehicle representation and on the other hand, the importance of selecting the information according to a quality criterion to improve the re-identification performance. It also identifies the limitations of the proposed approach.



## **Acknowledgments**

I would like to express my sincere gratitude to my supervisor Professor Benoît Macq and assistant Dani Manjah for the guidance and valuable advice they provided during the year.

In addition I am extremely grateful to Benoit Maison who truly provided useful advice and feedback, but also my father Philippe Gérin and Laure-Anne Dumortier who assisted me collecting the data.

I finally address my special thanks to François-Xavier Inglese for his patience and willingness to help me find solutions to my problems.

# Summary

|                                                                     |           |
|---------------------------------------------------------------------|-----------|
| <b>Introduction</b>                                                 | <b>1</b>  |
| <b>1 Vehicle re-identification in computer vision</b>               | <b>3</b>  |
| 1.1 Problem statement . . . . .                                     | 3         |
| 1.2 Machine learning and deep learning for vehicle reID . . . . .   | 4         |
| 1.3 Other vehicle reID algorithms . . . . .                         | 6         |
| 1.3.1 Local features . . . . .                                      | 6         |
| 1.3.2 Unsupervised learning . . . . .                               | 7         |
| 1.3.3 Attention mechanism . . . . .                                 | 8         |
| 1.4 Similarity learning . . . . .                                   | 9         |
| 1.4.1 Contrastive loss and siamese network . . . . .                | 9         |
| 1.4.2 Triplet loss and triplet network . . . . .                    | 11        |
| 1.5 Global comparison of the different reID methods . . . . .       | 13        |
| <b>2 Multi-camera scenario for vehicle reID</b>                     | <b>15</b> |
| 2.1 Parking lot scenario description . . . . .                      | 15        |
| 2.2 Available data . . . . .                                        | 16        |
| 2.3 Critical aspects . . . . .                                      | 18        |
| <b>3 Siamese network for vehicle reID</b>                           | <b>20</b> |
| 3.1 Approach and motivation . . . . .                               | 20        |
| 3.2 Architecture design . . . . .                                   | 20        |
| 3.3 Network training . . . . .                                      | 23        |
| <b>4 Multi camera re-identification</b>                             | <b>24</b> |
| 4.1 Multiple vehicle detections within a single viewpoint . . . . . | 25        |
| 4.2 Stacking multiple vehicle representations . . . . .             | 26        |
| 4.3 Estimation of the quality of an image . . . . .                 | 27        |
| 4.4 Database design . . . . .                                       | 29        |
| 4.5 Comparing two multi-view representations . . . . .              | 29        |
| 4.6 Assignment to the database . . . . .                            | 30        |
| 4.7 Full multi-camera vehicle reID algorithm . . . . .              | 31        |
| 4.7.1 Database initialization . . . . .                             | 31        |
| 4.7.2 Iteration phase . . . . .                                     | 32        |
| <b>5 Presentation of results</b>                                    | <b>34</b> |
| 5.1 Siamese network performance . . . . .                           | 34        |
| 5.2 Multi-camera reID algorithm performance . . . . .               | 36        |
| <b>Conclusion</b>                                                   | <b>40</b> |

# Glossary

|        |                                                      |                                                                                                                                                                                                                                                 |
|--------|------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CBLR   | Convolution + batch normalization + leaky ReLU block | <a href="#">21</a> , <a href="#">22</a>                                                                                                                                                                                                         |
| CNN    | Convolutional neural network                         | <a href="#">5</a> , <a href="#">6</a> ,<br><a href="#">15</a> , <a href="#">18</a> ,<br><a href="#">20</a> , <a href="#">21</a> ,<br><a href="#">23</a> , <a href="#">34</a>                                                                    |
| DL     | Deep learning                                        | <a href="#">5</a>                                                                                                                                                                                                                               |
| GAN    | Generative adversarial network                       | <a href="#">7</a> , <a href="#">8</a>                                                                                                                                                                                                           |
| GPU    | Graphics processing unit                             | <a href="#">20</a> , <a href="#">23</a>                                                                                                                                                                                                         |
| HOG    | Histogram of Oriented Gradients                      | <a href="#">5</a>                                                                                                                                                                                                                               |
| LBP    | Local Binary Patterns                                | <a href="#">5</a>                                                                                                                                                                                                                               |
| mAP    | Mean Average Precision                               | <a href="#">6–8</a> , <a href="#">11</a> ,<br><a href="#">13</a> , <a href="#">36</a>                                                                                                                                                           |
| ML     | Machine learning                                     | <a href="#">4</a> , <a href="#">5</a>                                                                                                                                                                                                           |
| NMS    | Non maximum suppression                              | <a href="#">25</a>                                                                                                                                                                                                                              |
| NN     | Neural network                                       | <a href="#">5</a>                                                                                                                                                                                                                               |
| PGAN   | Part-guided attention network                        | <a href="#">8</a> , <a href="#">9</a>                                                                                                                                                                                                           |
| RAM    | Random access memory                                 | <a href="#">20</a>                                                                                                                                                                                                                              |
| reID   | Re-identification                                    | <a href="#">1</a> , <a href="#">3–8</a> ,<br><a href="#">11</a> , <a href="#">13–</a><br><a href="#">20</a> , <a href="#">24–</a><br><a href="#">27</a> , <a href="#">29–</a><br><a href="#">33</a> , <a href="#">35–</a><br><a href="#">40</a> |
| ReLU   | Rectified Linear Unit                                | <a href="#">6</a> , <a href="#">21</a> ,<br><a href="#">22</a>                                                                                                                                                                                  |
| ResNet | Residual network                                     | <a href="#">6</a> , <a href="#">20</a>                                                                                                                                                                                                          |
| RGB    | Red Green Blue                                       | <a href="#">21</a> , <a href="#">40</a> ,<br><a href="#">41</a>                                                                                                                                                                                 |
| ROI    | Region of interest                                   | <a href="#">7</a> , <a href="#">28</a>                                                                                                                                                                                                          |
| SIFT   | Scale-Invariant Feature Transform                    | <a href="#">5</a> , <a href="#">11</a>                                                                                                                                                                                                          |

# Introduction

Object identification techniques have taken a lot of importance in the computer vision realm, especially thanks to the progress in deep learning. Indeed, deep learning has unlocked the previous state-of-the-art boundaries which were achieved by classical algorithms and the progress in the domain has underlined the great potential of this approach as a response to human needs.

Figure 1 shows the technology readiness level of artificial intelligence in 2020. As one could observe, deep learning, machine learning and computer vision are generating many expectations which is reflected by the number of published papers in the scientific literature. These three technologies that are the foundations of this thesis should reach their plateau of productivity in less than five years according to same predictions. The hype around these domains can be explained by the increasing need to automatically process a large amount of data and information.

In this thesis the subject of re-identification applied to vehicles is studied to identify a use case where this branch of computer vision can be improved to increase the performance of the algorithms and again push the horizon of possibilities further beyond.

The objective of this thesis is on one hand to design a vehicle re-identification algorithm which can benefit from multiple cameras to detect and track several vehicles on a time-varying parking lot. On the other hand the interest of using several cameras to increase the performance of a vehicle re-identification algorithm is studied.

To that end, a siamese deep neural network is designed and trained as a tool to compare vehicle images and estimate how similar the presented vehicles are. Furthermore a handcrafted dataset is constructed and annotated to perform several tests and quantify the quality of the developed re-identification algorithm.

To justify the design choices, vehicle re-identification literature is reviewed to identify the different existing methods to recognize vehicle identities and list the advantages and drawbacks of the main families of approaches.

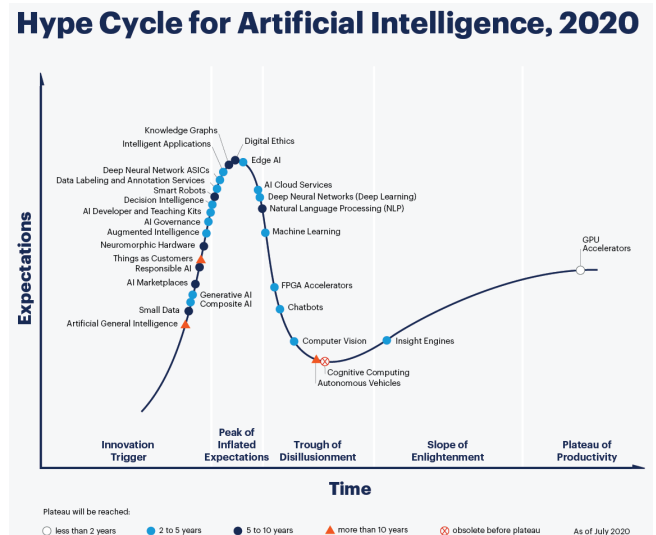


Figure 1: Gartner hype cycle [1]

In order to meet these objectives, the writing is organized as follows:

- Chapter 1 first presents the vehicle re-identification principle and the current limitations of the actual approaches. In a second time the different kinds of methods are discussed with specific examples to illustrate their purpose. A specific family of methods is pushed forward during the literature review since it offers many advantages related to the thesis objectives. Finally a summarized comparison of the discussed methods highlights the qualities and drawbacks of each approach.
- Chapter 2 precisely settles the context and specifications of a multi-camera scenario. The parking lot scenario for vehicle re-identification with several cameras is described, followed by a discussion about the available data to train and experiment a model. The handcrafted dataset is also presented and its content is analyzed. The last part of this chapter aims to identify the critical aspects of a multi-camera vehicle re-identification scenario in order to make a list of specifications for the algorithm.
- Chapter 3 focuses on the architecture design and training of a siamese deep neural network. The interest and choice for such an architecture is discussed and justified at first. Then the technical and mathematical aspects of the model at every level are established. Finally the preparation of the training set and the training phase are explained.
- Chapter 4 refers to the multi-camera vehicle re-identification algorithm. It describes and justifies each element and step that are part of the whole algorithm. It first deals with the concept of multi-view representation of a vehicle and how it is assembled in practice. Moreover the concept of image quality is analyzed and specified in order to design an image quality metric since the algorithm treats the information according to its relative quality. Before presenting the algorithm, the method to compare multi-view representations of vehicles is tackled. Finally the chapter ends with the presentation of both the initialization and the iteration phases of the algorithm.
- Chapter 5 aims to explain and quantify the performance of what has been developed in this thesis with also pertinent discussion and interpretation. The first part characterizes the siamese network over both the same kind of data as for the training phase and the handcrafted dataset. The second part focuses on the whole re-identification algorithm at several levels and brings a discussion about the impact of each parameter.

The thesis is concluded with a summary of what have been done during the thesis, the main results and discussions. It also identify several ways of improvement.

## Chapter 1

# Vehicle re-identification in computer vision

Object re-identification (**reID**) in computer vision is the process of associating several images or videos of given objects taken from different angles and camera viewpoints. It is mainly applied to person re-identification, but it can also be used to recognize vehicles or basic objects. This chapter aims to provide to the reader a glimpse of the current situation in the vehicle re-identification realm. In section 1.4, similarity learning, a more specific deep learning technique is detailed since it will be used for application presented in chapters 3 and 4.

### 1.1 Problem statement

Vehicle **reID** problem is commonly described as an instance retrieval of given query images (i.e., the vehicles instances one would like to recognize) among a set of other vehicle images (i.e., the gallery). Figure 1.1 depicts an example of vehicle **reID** results. Each query image is compared with many vehicle images within the gallery captured by several cameras. The matching images are then ranked according to their confidence score provided by the **reID** algorithm in use.

To this day, vehicle re-identification remains a difficult computer vision task despite many advances in the domain. Indeed, vehicles present much less appearance diversity than humans who can be distinguished according to their clothes, hair, sizes and proportions. When comparing vehicles, the diversity drastically drops since the color distribution steers toward shades of gray (figure 1.2) and in addition the vehicle shapes are similar due to the limited number of automobile brands and manufacturers [4]. Overall, we can list the most problematic challenges while addressing a vehicle **reID** task:

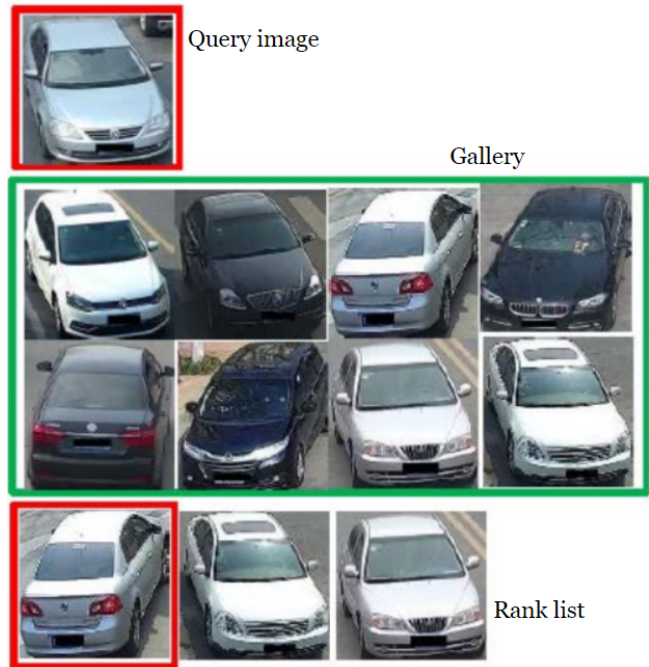
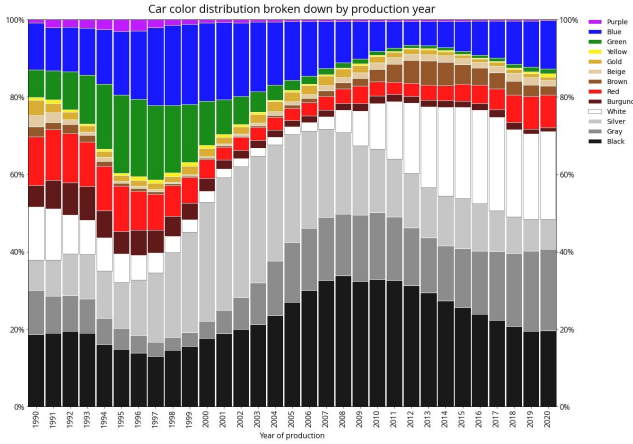


Figure 1.1: Vehicle **reID** ex. (VeRi dataset) [2]

- Lack of diversity: as explained, the poor diversity of vehicle shapes and colors can sometime make the comparison task extremely difficult.
- Illumination changes: due to the different lighting conditions of the cameras, the photographed vehicle color feature could vary greatly from one camera to another.
- Occlusions: an image could only partially represent the targeted vehicle if taken in dense traffic, leading to loss of information or evenly misinformation if the occlusion is due to another vehicle.
- Scale and resolution changes: vehicles can be captured in the camera near-sight as well as in its far-sight, leading to great variations of image resolutions.



**Figure 1.2:** Vehicle color distribution [3]

PKU Vehicle-ID provides a great amount of classes and images, but has been recorded by only two cameras observing a crossroad leading to few representation of each vehicle class. VERI-Wild dataset offers images from a complex unconstrained capture conditions by 174 cameras in an urban district and is well suited for smart-city surveillance applications. The last discussed dataset is CityFlow, which can be compared with VERI-Wild in the sense that it also focuses on an urban scenario. The advantage of CityFlow compared to VERI-Wild is the number of images per vehicle, that is significantly larger.

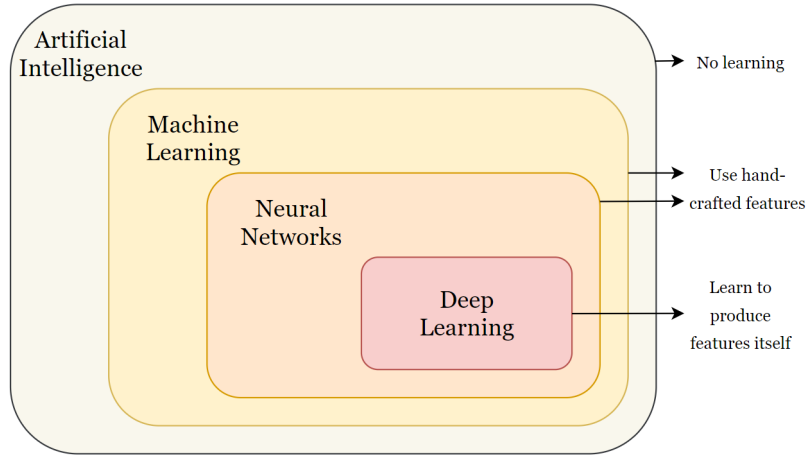
|         |                   | Training diversity |                  |          |
|---------|-------------------|--------------------|------------------|----------|
|         |                   | #Vehicle images    | #Vehicle classes | #Cameras |
| Dataset | VeRi-776[5]       | 37, 781            | 576              | 20       |
|         | PKU Vehicle-ID[6] | 221, 763           | 26, 267          | 2        |
|         | VERI-Wild[7]      | 416, 314           | 40, 671          | 174      |
|         | CityFlow[8]       | 36, 935            | 333              | 40       |

**Table 1.1:** Vehicle reID dataset comparison

## 1.2 Machine learning and deep learning for vehicle reID

The existing algorithms can be split into two main categories before branching into sub-categories. On one hand, we have the traditional machine learning-based algorithms and on the other hand the deep learning realm. The situation is represented on figure 1.3. ML-based methods learn to solve the reID task with handcrafted features, that is, it does not take the raw images as input but rather uses

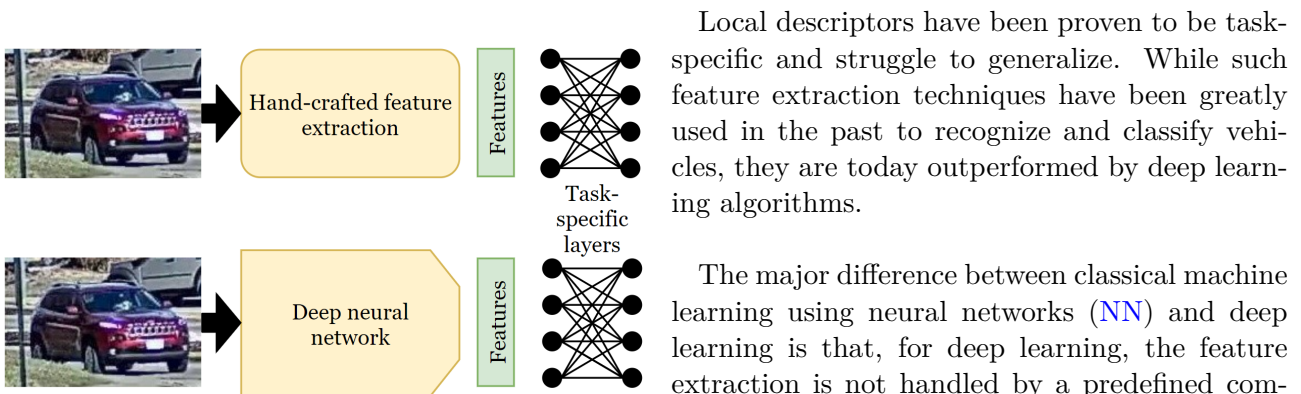
quantities produced by what is called feature extraction, which is responsible for refining and cleaning the raw data.



**Figure 1.3:** Visual representation of ML and DL subsets

Feature extraction in computer vision is often addressed by local descriptors. A local descriptor produces a feature for a given pixel and its neighborhood, it could be an information about the local shape, color or texture. Three methods are commonly used:

- Local Binary Patterns (LBP) [9] is a simple descriptor technique used to describe local texture by comparing each pixel of an image to its neighboring pixels and producing a binary number.
- Histogram of Oriented Gradients (HOG) [10] is used for target detection and recognition. It is based on local direction of the gradients within a patch of pixels and focuses on edge information.
- Scale-Invariant Feature Transform (SIFT) [11] is resistant to rotation, scale variation and brightness changes. It focuses on the local appearance of the image.



**Figure 1.4:** Structure of NN-based approach

Local descriptors have been proven to be task-specific and struggle to generalize. While such feature extraction techniques have been greatly used in the past to recognize and classify vehicles, they are today outperformed by deep learning algorithms.

The major difference between classical machine learning using neural networks (NN) and deep learning is that, for deep learning, the feature extraction is not handled by a predefined computation tool but instead by another NN with way more layers (figure 1.4) which is trained to learn by itself how to extract pertinent high-level

semantic information from its input. The extracted feature vector is then passed through task-specific layers, also known as the network *head*. For example, in case of vehicle classification problem, the *head* is often designed to output a probability vector of size equals to the number of different classes based on the information provided by the feature extraction step.

For spatio-temporal structured input data (e.g., images or audio sequences), deep convolutional neural networks (CNN) have proven to be a superior network architecture in term of training resources,



performance and avoiding overfitting compared to classical fully connected networks [12].

One of the main advantages of deep CNN is that, since they are well designed to generalize, they can be trained on one dataset to learn how to produce high-level features of objects within an image and are thus often called *backbones* in the sense that they can be detached from their original *head* and combined with another *head*. Thanks to *backbone* replacement, the learning phase is much quicker since the new pre-trained *backbone* has already seen how to construct useful features, and thus does not need to learn from scratch. One can cite the most popular *backbones* in term of performance:

- AlexNet [13] is composed of 5 convolutional layers and 3 fully connected layers, it has around 60 million parameters. It is also the first CNN to use rectified linear units (ReLU [14]) as activation functions which are solving the vanishing gradient problem [15] during training. AlexNet is used in object detection models such as RCNN [16].
- VGG-16 [17] is composed of 13 convolutional layers and 3 fully connected layers with ReLU activation once again. It has about 138 million parameters and is also used in object detection problems.
- GoogLeNet [18] is a smaller network using a brand new notion known as blocks of inception allowing multi-scale convolutional transformations. It has 22 layers but only 5 million parameters and is most suited for real-time object detection such as YOLO [19].
- The ResNet [20] architecture uses batch normalization [21] on one hand and residual connections through the network on the other hand. Both methods accelerate the training phase and allow deeper CNNs with more layers, the number of layers for ResNet architectures ranging between 18 and 152. It is mainly used in object detection.

### 1.3 Other vehicle reID algorithms

While similarity learning is discussed in section 1.4, this section presents the advantages and drawbacks of other reID methods. To quantify the performance, the mean Average Precision (mAP) metric is used. The average precision measures how well a model is able to compare a single query image with the gallery. Mean average precision does the same but is averaged over all query images.

#### 1.3.1 Local features

The idea is that differences between two vehicles with similar appearance is hiding in local details such as the wheels, the brand logo or the headlights. The most common methods are using key point location and region segmentation. The paper [22] proposes to compute local orientation invariant features at 20 key points on the vehicle in order to be aligned with other vehicles. Thanks to a Stacked Hour-glass Network-like design (used in human pose estimation task [23]), they are able to identify the 20 key points and estimate the vehicle orientation in order to remove occluded key points. The global image and every local key points are

passed through feature extraction CNN to produce several feature vectors. The next step is to aggregate all the feature vectors with pooling and fully connected layers to obtain a single orientation invariant feature in charge of representing the vehicle. During the retrieval process of a vehicle with a gallery, they use what they call the appearance distance  $D_a$  and the spatio-temporal distance  $D_s$  in order to take a decision.  $D_a$  is based on comparison of two feature representations and  $D_s$  uses



Figure 1.5: Vehicle annotation [22]

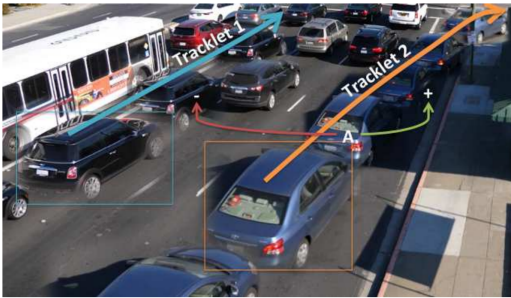
additional camera information such as capture time stamp difference and camera location distances. During the testing phase, they achieve a 51.42% mAP score on VeRi-776 dataset. While the feature representation is robust, the spatio-temporal information are greatly contributing to the performance and is well suited for city-surveillance scenario. However, spatio-temporal information is not always available in other applications.

One can also cite other similar works using local features, such as [24] where again local features are extracted from regions of interest (ROI) before being combined into a structural feature of the vehicle, or [25] where they propose a multi-region model to extract both global features and local features.

To summarize, local feature methods are able to capture unique visual clues to reduce the errors due to vehicles with similar appearances. However, these methods require the extraction of many local features which can be an issue regarding the computational resources and struggle to compare local features when the camera view points are too different.

### 1.3.2 Unsupervised learning

As explained in section 1.1, the lack of annotated data for vehicle reID is a serious issue and is currently limiting the generalization capabilities of the algorithms. A solution is to use unsupervised learning techniques in order to increase the amount of available data and therefore alleviate the need for costly annotated data.



**Figure 1.6:** Vehicle tracklets for automatic annotation [26]

In [26], they show how vehicle tracking can be exploited to generate a weakly labeled dataset to train a deep neural network in order to perform reID task. Based on traffic videos (figure 1.6), a detector extracts all the vehicles within the image and assigns each detection to an existing or a new tracklet (i.e., a time-sorted image collection of a same vehicle). Based on images from different tracklets, they are able to train a triplet network architecture (discussed in section 1.4) with the initially unsupervised data. Then, a matching strategy aims to re-identify vehicles between the different videos. However, they did not provide any metric to compare with other reID algorithms, but their idea highlights the great opportunity to use unlabeled data.

The generative adversarial network (GAN [27]) is another tool in the realm of unsupervised data usage and has proven to be useful for data generation by minimizing the gap between the synthetic and physical domains. The generative part of GAN learns to produce fake data as real as possible and the adversarial part, in competition, tries to determine if the produced data are real or fake. By doing so, the competition reaches an equilibrium with both sides doing the best they can.

In [28], they develop a perspective-aware generative model (PAGM) to generate new images by reconstructing a different view point of a given input vehicle with a targeted pose (figure 1.7). They prove that their generative model can produce realistic images by preserving the local details from the source to the target.

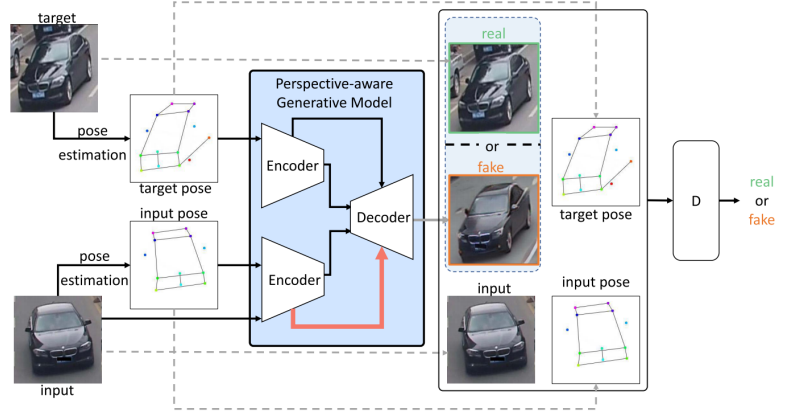


Figure 1.7: Vehicle image generation in [28]

In [29] they also explore data augmentation with GAN to build new vehicle images by modifying the vehicle color and texture. In addition, they extract the vehicle within an image to modify the background in order to artificially increase the data diversity. Thanks to these techniques they increased the mAP score by 3.03%, highlighting the interest to produce synthetic data.

### 1.3.3 Attention mechanism

This category of techniques aims to mimic the attention mechanisms of the human. For example, when a human tries to identify a vehicle, he will first examine the high-level attributes such as the brand or the global shape, and then identify low-level attributes such as windshield stickers. In [30], they propose a Recurrent Neural Network-based Hierarchical Attention (RNN-HA) classification model for vehicle reID. It consists of three cascaded models that imitates the top-down decision making mechanism of a human (figure 1.8). They reach a mAP score of 56.80% on the VeRi dataset.

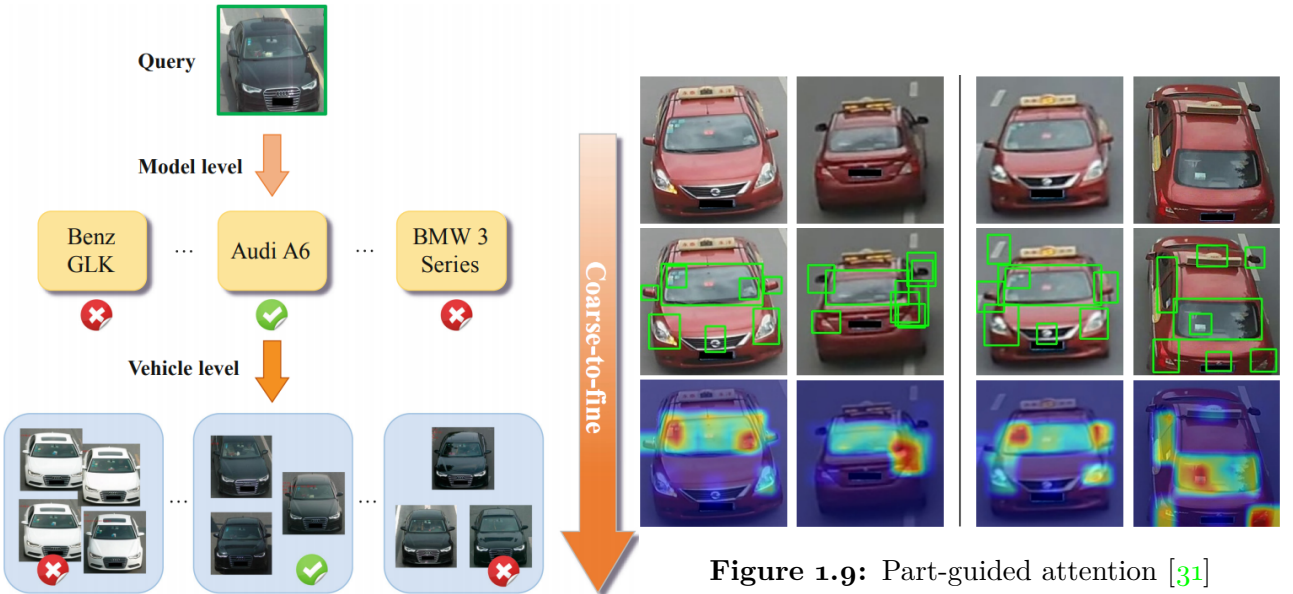


Figure 1.8: Top-down attention [30]

In [31], they introduce a Part-Guided Attention Network (PGAN) to focus on prominent part regions and effectively combine the global and local information for discriminative feature learning. The PGAN detects the salient regions regardless of the vehicle identity (figure 1.9) after which each region is treated by a module in charge of giving weights in order to discard distracting regions such as the

background and other irrelevant information and giving more importance to the useful discriminative regions. In the same philosophy of local features techniques (subsection 1.3.1), they aggregate the global appearance and local features together before ranking. They thus combine both bottom-up and top-down attention mechanisms with the PGAN and the weighting module respectively.

However, attention mechanism techniques have been proven to focus on regions at the expense of the differences of finer pixel level. Still, this kind of approach can be used to verify if a model is focusing on pertinent regions of the image. In [32] they present Grad-CAM, a visualization technique allowing to highlight in the feature maps which regions of the input image are producing the higher feature values. It is useful to verify if the model focuses on salient and discriminative elements of the vehicle (figure 1.10).



Figure 1.10: Visualization of feature maps in [29] using Grad-CAM

## 1.4 Similarity learning

Similarity learning, also named distance learning or metric learning, is a method of feature space transformation which aims to attract the feature representation of vehicles with the same label and to repel vehicles belonging to different classes. Therefore, the network must somehow learn the key distinguishing features to discriminate vehicles from different classes. Similarity learning is mostly about the design of the loss function to train the network. The two most popular losses are the contrastive loss [33] and the triplet loss [34]. These loss functions are comparing the feature vectors of respectively 2 and 3 input images in order to actuate the feature extraction *backbone* during training.

### 1.4.1 Contrastive loss and siamese network

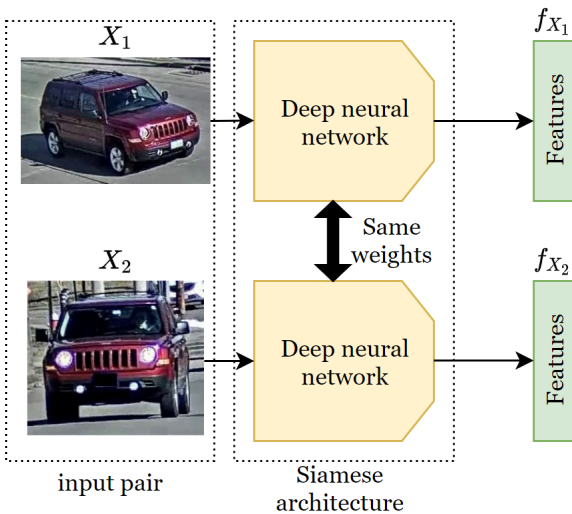


Figure 1.11: Siamese network

The contrastive loss is used to train what is called a siamese network. A siamese network (represented on figure 1.11) is composed of two deep neural networks which are sharing the same weights and architecture. During the training phase, their weights are thus forced to be updated with same values.

It is mainly used to measure the similarity between image pairs by comparing their feature representation, that is, by estimating how much the input images belong to the same class or not. In addition, it has the great advantage to train on pairs of images in order to learn view-point invariant feature representation for the vehicles.



In term of notations, the siamese network is given an image pair  $\{X_1, X_2\}$  and outputs a pair of feature vectors  $\{f_{X_1}, f_{X_2}\}$ . The one-to-one relationship between an image  $X_i$  and its feature representation  $f_{X_i}$  is a complex non-linear function and is thus written as a black-box relation in equation 1.1:

$$f_{X_i} = G_W(X_i) \quad (1.1)$$

Where  $G_W$  denotes the deep network function with the weight set  $W$ . In addition, the Euclidean distance  $D_W$  between two feature representation is defined in 1.2:

$$D_W(X_1, X_2) = \|G_W(X_1) - G_W(X_2)\|_2 = \|f_{X_1} - f_{X_2}\|_2 \quad (1.2)$$

The contrastive loss  $L^{\text{cont}}$  [33] for a sample  $i$  composed of two images and a target  $(X_1, X_2, Y)^i$  is defined in 1.3:

$$L^{\text{cont}}(W, (X_1, X_2, Y)^i) = (1 - Y) \underbrace{\frac{1}{2}(D_W)^2}_{L_S(W, (X_1, X_2, Y)^i)} + (Y) \underbrace{\frac{1}{2}(\max(0, m - D_W))^2}_{L_D(W, (X_1, X_2, Y)^i)} \quad (1.3)$$

where:

- $Y$  is the binary target and is 0 if the two images are from the same class and is 1 otherwise.
- $L_S$  is the partial loss function for similar labels.
- $L_D$  is the partial loss function for different labels.
- $m$  is a positive margin.

And the global contrastive loss  $\mathcal{L}^{\text{cont}}$  of  $P$  samples is simply the sum of all sample losses:

$$\mathcal{L}^{\text{cont}}(W) = \sum_{i=1}^P L^{\text{cont}}(W, (X_1, X_2, Y)^i) \quad (1.4)$$

As one can observe, equation 1.3 represents a loss with two distinct modes depending on the sample ground truth. If the two images of the  $i^{\text{th}}$  sample have the same label ( $Y = 0$ ), then the loss is reduced to the squared Euclidean distance between their feature representations. On the contrary, the mode for different labeled images ( $Y = 1$ ) reduces the loss to the measure of how close the feature vectors are in the feature space. The margin  $m$  indicates the distance from which the loss is considered null since the feature vectors are spaced enough.

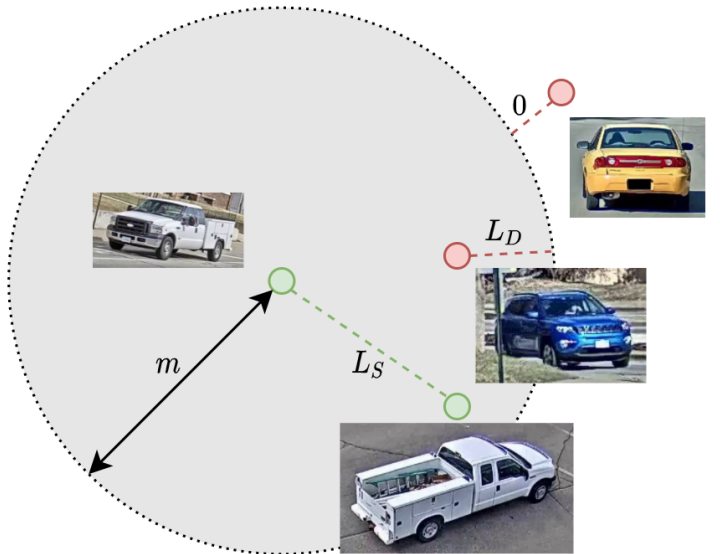
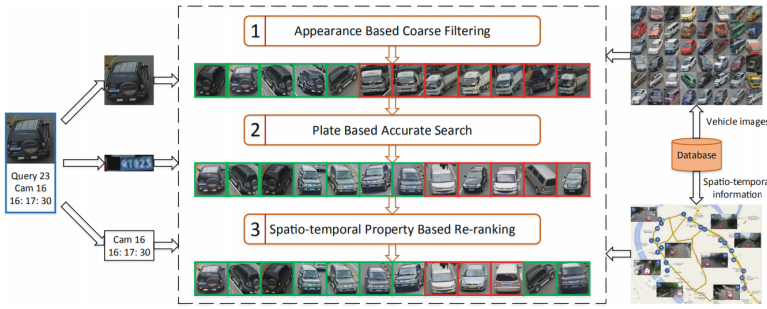


Figure 1.12 depicts an example of the contrastive loss in the feature space. When the

**Figure 1.12:** Visual representation of contrastive loss in feature space

distance between two images with different labels is greater than the margin  $m$ , the loss is null.

Performing a gradient descent with the contrastive loss will progressively decrease  $L_S$  between the feature vectors from the same label and try to repel the feature representation of different vehicles to the margin distance. Ideally, images under the same label are gathered into distinct clusters, and all the clusters are separated by at least the margin distance.

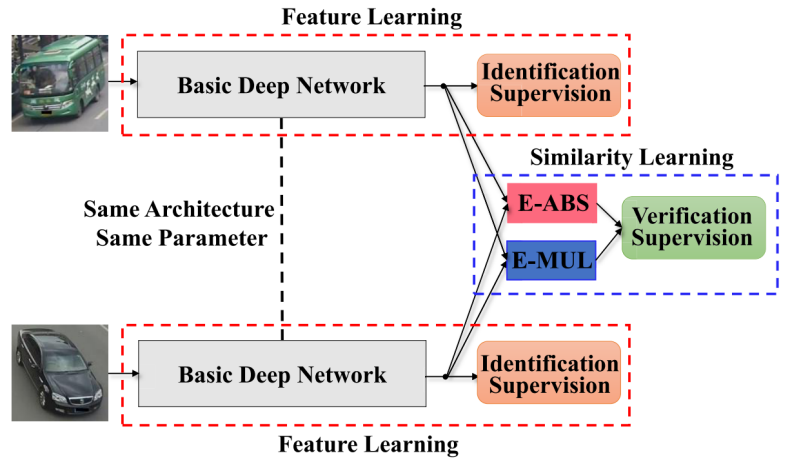


**Figure 1.13:** PROVID with siamese network [35]

Many reID algorithms use the siamese architecture. In [35] they propose PROVID, a three staged reID process which first performs a coarse filtering with SIFT descriptors in order to remove vehicles with different color, texture and shape from the gallery. Then, a siamese network is used to compare the query license plate with the remaining vehicles within the gallery. It finally re-ranks the gallery by using the camera spatio-temporal knowledge to remove the images which

are too far in time or in space. PROVID achieves a mAP score of 53.42% on VeRi-776 and they point out the interest of this low-computation and fast method for real-time urban scenarios.

In [36] they introduce a siamese neural network trained under the joint identification and verification supervision (figure 1.14). The identification part is used to learn deep features for vehicle images while the verification part aims to learn a similarity metric to compare the feature representations.

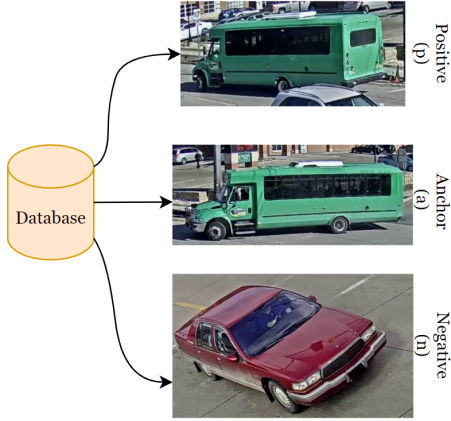


**Figure 1.14:** Joint identification and verification training [36]

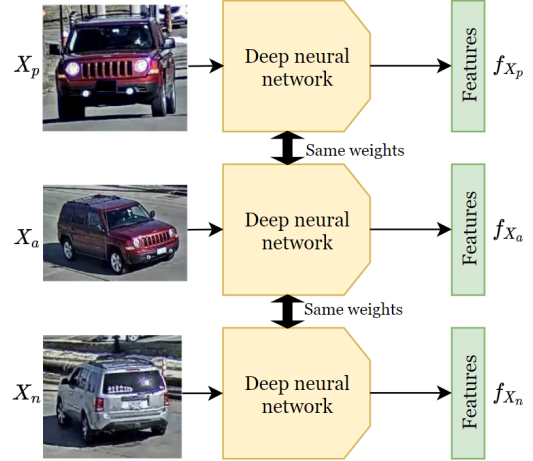
Both supervisions are applied together by the construction of a loss function composed of three terms. Two terms are related to the identification part of the two input images and use the softmax function [37]. The similarity loss compares the absolute difference and the product between the two feature representations. The three contributions are then summed up with different weights. They achieved a mAP score of 53.53% on VeRi-776 and showed that their method outperforms most of the state-of-the-art reID algorithms without using any license plate information or camera time stamp/position annotation.

#### 1.4.2 Triplet loss and triplet network

With regard to the contrastive loss, the triplet loss [34] is built on the comparison of three input images. Based on an image called an anchor ("a" in short), two images with the same and different vehicle labels, respectively the positive (p) and the negative (n), are extracted from the database (figure 1.15).



**Figure 1.15:** Image triplet description



**Figure 1.16:** Triplet network structure

The architecture of a triplet network is depicted in figure 1.16. Three deep neural networks with the same architecture are used, sharing the same weights just like the siamese architecture. The image triplet  $\{X_p, X_a, X_n\}$  is producing the feature vectors  $\{f_{X_p}, f_{X_a}, f_{X_n}\}$  when forwarded through the network. Using the same notations and conventions as equations 1.1 and 1.2, the optimization target for the  $i^{\text{th}}$  image triplet is:

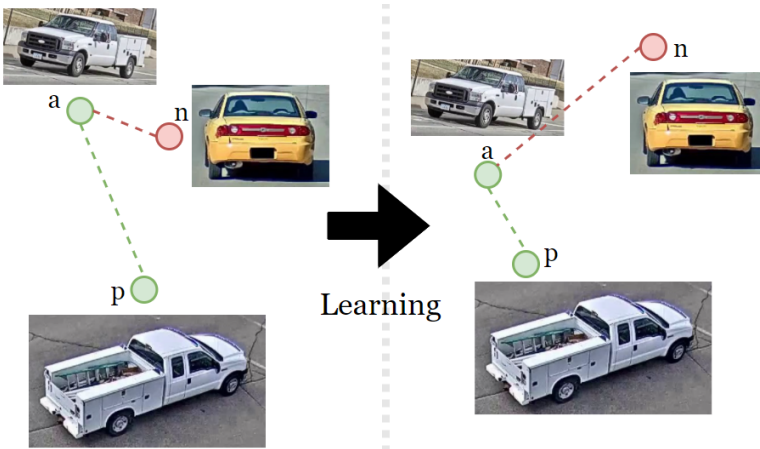
$$L^{\text{trip}}(W, (X_p, X_a, X_n)^i) = \frac{1}{2} [(D_W^{a,p})^2 - (D_W^{a,n})^2 + \alpha]_+ \quad (1.5)$$

where:

- $D_W^{a,p}$  is the Euclidean distance between the anchor and positive image features.
- $D_W^{a,n}$  is the Euclidean distance between the anchor and negative image features.
- $\alpha$  is the violate margin.
- $[\cdot]_+ = \max(0, \cdot)$  is the hinge function.

And the global triplet loss  $\mathcal{L}^{\text{trip}}$  of  $P$  samples is the sum of all sample losses:

$$\mathcal{L}^{\text{trip}}(W) = \sum_{i=1}^P L^{\text{trip}}(W, (X_p, X_a, X_n)^i) \quad (1.6)$$



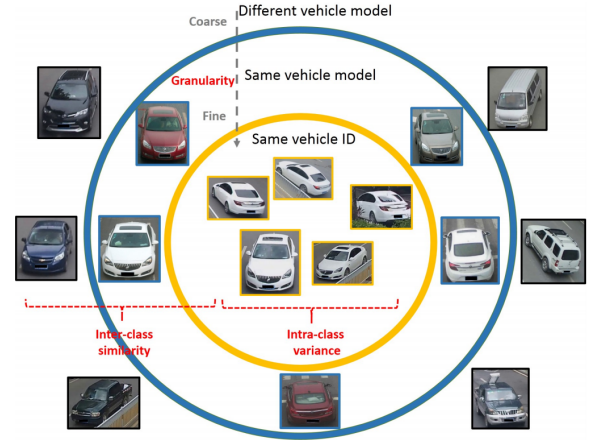
**Figure 1.17:** Triplet loss learning in feature space

The goal of the triplet loss is to attract the anchor and positive feature representations in the feature space and repel anchor and negative representations simultaneously. Reducing the loss sequentially will form clusters of vehicles with the same label in the feature space (figure 1.17).

If  $D_W^{a,n} + D_W^{a,p} \leq \alpha$ , the loss is null due to the hinge function. In that case, the feature vectors are considered to be sufficiently separated, which avoids spending too much resources into

pushing the negative image as far as possible from the anchor.

In [38], they designed the Group Sensitive Triplet Learning (GTSE) to recognize and retrieve vehicles. During learning, the network handles the large inter-class similarity and the intra-class variance of vehicle reID problem by clustering the feature representation and by building the triplet by sampling the dataset into both the inter-class similarity and intra-class variance areas (figure 1.18) in order to force the deep neural network to focus on discriminative elements and learn fine-grained features. That is, the model is trained to separate vehicles of different car models on one hand and to perceive fine differences with different vehicles from the same car model on the other hand.



**Figure 1.18:** Group-sensitive model in [38]

Their method reaches a mAP score of 59.47% on VeRi-776, and has proven to deal effectively with the intra-class variance in learning representation.

In [39] they tackle the vehicle reID problem by using a more constrained triplet loss to re-enforce the results. To do so, they supervise the learning with the triplet loss and a classification-oriented loss jointly. In addition, they propose to use a new triplet sampling method: instead of random sampling of triplets, they form the learning triplet in a way to handle both the intra-class variance and inter-class similarity in an efficient way. Their algorithm achieves a mAP score of 58.78% on VeRi-776.

## 1.5 Global comparison of the different reID methods

The previous sections presented the most prominent approaches to handle a vehicle reID problem, with more mathematical details for similarity learning (1.4) since it will be useful in chapter 3.

The table 1.2 summarizes the benefits and drawbacks of each method. It is important to note that each algorithm is in fact a combination of several approaches. For example, the unsupervised learning papers in subsection 1.3.2 propose data augmentation techniques, which can be used as a complement with annotated data.

According to the reviewed papers, dealing explicitly with the inter/intra-class similarity issue is an approach that brings interesting results since it is the major issue when treating a vehicle reID problem. To this end, increasing the representation and therefore the quantity of captured information about a vehicle might be a solution to increase the performances of a vehicle reID algorithm by considering a vehicle as a more complex object with more degrees of freedom that a single image cannot always provide.

Similarity learning methods were put forward in this review since the relative robustness to camera viewpoint variations is able to counteract situation where the two-dimensional images of a given vehicle are showing strongly different salient elements.

Combining these two ideas leads to the subject of this thesis: proposing and designing a multi-camera vehicle reID algorithm that can take advantage of several representations of the same vehicle by comparing the different viewpoints accurately.



| Method                | Description                                           | Advantages/Disadvantages                                                                                                                                                                            |
|-----------------------|-------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Local features        | Comparison of key point areas and image segmentation. | Capture unique visual clues.<br>Feature extraction is slow.<br>Struggle with view point variation.                                                                                                  |
| Unsupervised learning | Use unlabeled data.                                   | Attempt to deal with the lack of annotated data.<br>Create artificially diversity.<br>Expensive in resources.                                                                                       |
| Attention mechanism   | Imitate human attention processing.                   | Learn to focus on saillant areas.<br>Combine distributed low-level information.<br>Annotated dataset with region-labelling are not carrying enough diversity.<br>Sensitive to background variation. |
| Similarity learning   | Feature space transformation and clustering.          | Relatively robust to camera view-point variation.<br>Learning phase also cluster the feature space.<br>Training might be unstable due to data distribution.                                         |

**Table 1.2:** Comparison of vehicle re-identification methods

## Chapter 2

# Multi-camera scenario for vehicle reID

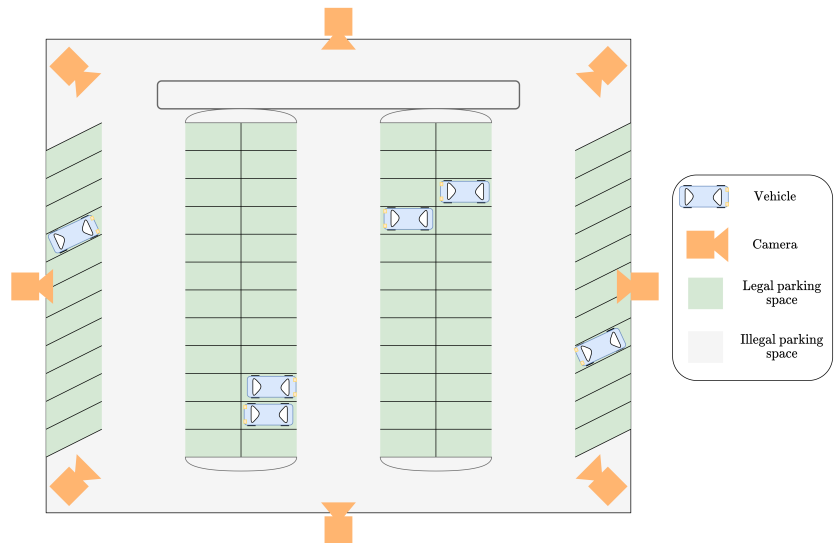
In chapter 1 it was seen that the major issue when dealing with vehicle reID task is the lack of visual diversity which is leading to inter-class similarity and intra-class variance. In order to push the algorithm performances forward, one could try to enhance the vehicle representation in some other way to counterbalance the current limitations.

In this chapter, a specific multi-camera scenario is presented and discussed. This particular case proposes room for improvement for vehicle reID research by exploring the benefit of using several cameras to observe the vehicles. The CNN design, the multi-cameras reID algorithm and the results will be presented in chapters 3, 4 and 5 respectively.

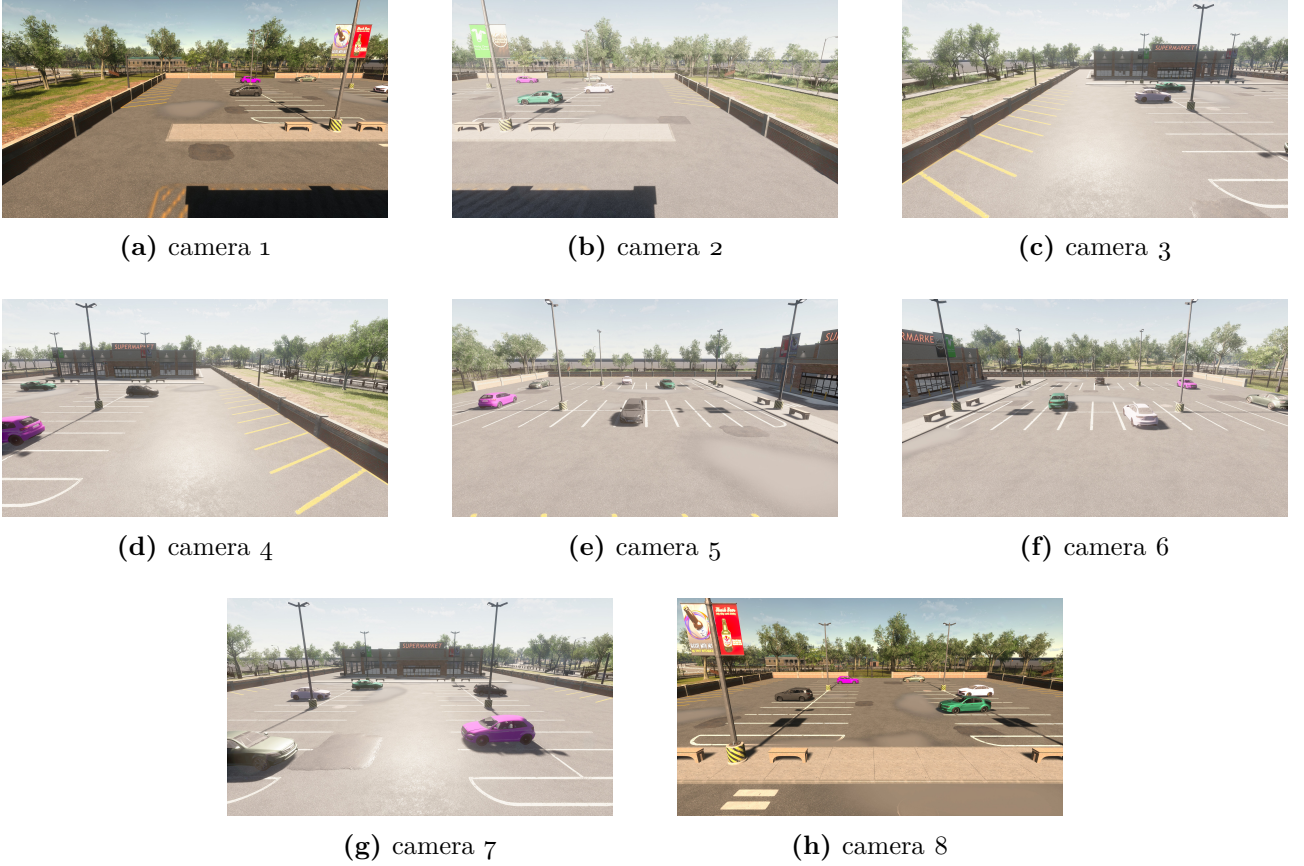
### 2.1 Parking lot scenario description

The general scenario is described by a car parking where several vehicles come, park themselves on parking spaces, remain still a certain amount of time and leave the environment to perhaps come and park again later on. For example it could be a store parking lot, a carpooling parking or a parking reserved for company staff.

Figure 2.1 depict an example of such a scenario. At a given instant, several vehicles are parked on legal parking spaces with specific direction and position. The parking lot is observed by cameras (8 in this case). Each camera has its own position, orientation, field of view and other intrinsic features. All cameras are connected to a central computer which is responsible to manage the capture and synchronization time of the camera network. The key aspect of such a configuration is that each parking space has a probability to be observed by more than one camera, to a maximum of 8 in this case.



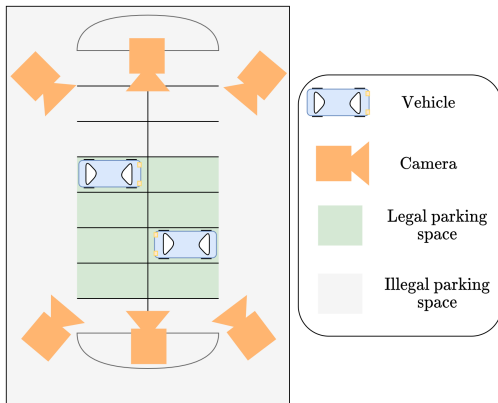
**Figure 2.1:** Example of parking lot scenario



**Figure 2.2:** Example of scene from the parking lot simulation

## 2.2 Available data

Up to this day, there is no publicly available dataset for vehicle **reID** on a parking lot. By following the described specifications, a parking simulation was first designed using Unreal Engine [40] in collaboration with an external developer [41]. The simulation implements the parking depicted on figure 2.1. As one can observe on figure 2.2 where the 8 viewpoints at time 0 are presented, the camera placement helps to observe specific vehicle multiple times with different angles and resolutions.



**Figure 2.3:** Hand-crafted scenario

While this synthetic dataset generation technique is promising, the actual version of this software is not able to produce enough diversity in term of vehicles and environment to be used as a training set. In addition, the lack of computational resources to produce high quality images is an important issue since the major drawback of using synthetic data is the gap between the physical and synthetic domains. However, this simulation tool is used to perform simple vehicle detection test and to provide visual examples.

In order to keep the diversity needed for vehicle **reID** training, it was decided to use the dataset

CityFlow[8] for the training phase. Furthermore, a handcrafted parking lot dataset was also recorded and annotated to test the algorithm in a real situation. Figure 2.3 depicts the configuration of the parking on which pictures were collected. For this dataset, 3 distinct cars are used to mimic a realistic behavior of drivers coming and leaving a small private parking lot.

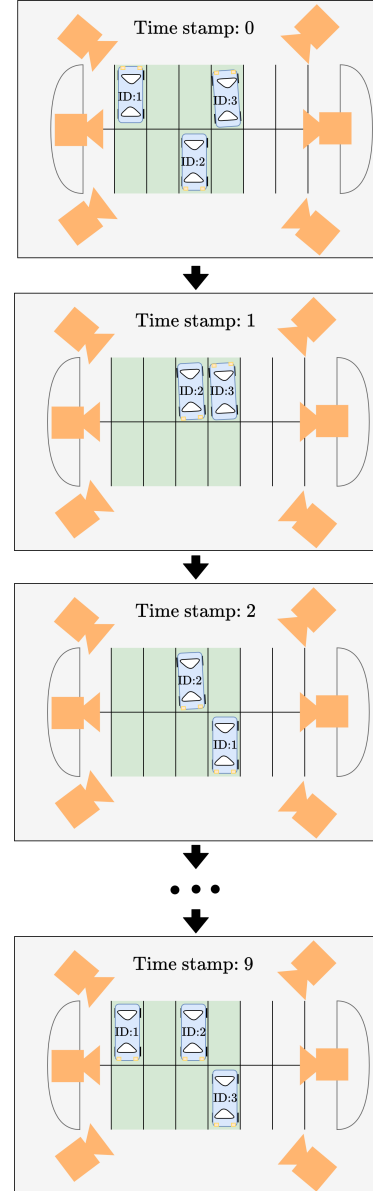
The scene is captured by 6 cameras giving different viewpoints of the parked vehicles. Figure 2.5 describes how the dataset was annotated: at a given instant, the cars are placed over the legal parking areas after what the scene is captured from 6 different locations around the area of interest. The frames are then cropped to separate the different vehicles detected among them and agglomerated to produce the multi-view images for each vehicle ID.

The dataset is composed of 10 scenes (figure 2.4), with at least one vehicle per scene for a total of 132 vehicle crops. Between two scenes, the cars can remain still or be moved away from the environment of interest in order to simulate a small parking for company staff used daily. The vehicle placement was done to bring variations of car orientations and to obtain several occlusions in order to mimic a real situation.

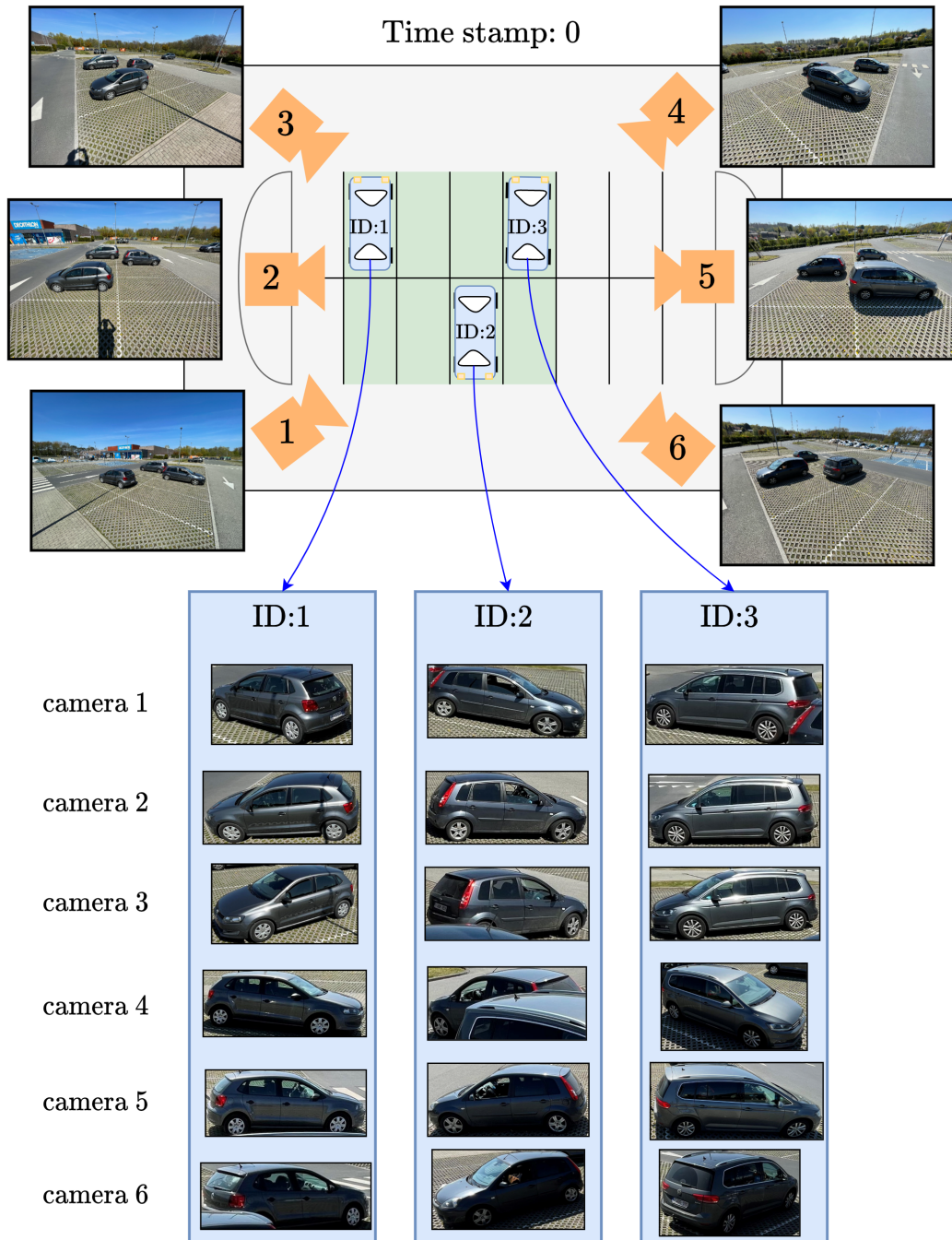
However, many aspects of a real case are not captured by this dataset:

- The diversity of the vehicle set, only 3 different cars are present in this dataset. Despite this aspect, the 3 cars have similar shapes and close-looking colors, which is useful to test a [reID](#) algorithm with strong inter-class similarities.
- The illumination conditions are roughly the same across the 10 scenes. Thus the dataset is not well suited to estimate the robustness to illumination changes of a [reID](#) algorithm.
- The cameras do not suffer from any visual deformations or other camera impairments. But it is important to note that it could be simulated if one would like to test an algorithm on such impairments.

Despite the mentioned issues of this hand-crafted dataset, it will be used to perform several tests of the vehicle [reID](#) algorithm proposed in chapter 5.



**Figure 2.4:** Scenes evolution



**Figure 2.5:** Hand-crafted dataset annotation at time stamp 0

## 2.3 Critical aspects

To allow the construction of a multi-camera [reID](#) algorithm, many aspects must be constrained to limit the implementation difficulties. Indeed, since a [CNN](#) will be used to perform the feature representation ([chapter 3](#)), all the real situations that could happen on a parking cannot be considered if a corresponding dataset does not exist. This is the case with the night, where the luminosity and color contrast perceived by cameras drastically drop. With these conditions, an algorithm trained on daytime dataset would produce poor results if it attempts to compare vehicle images from the day with images taken during the night. This is why the application should keep constant daytime



illumination to remain consistent with the illumination conditions of the training dataset.

Another important aspect is about the behavior of the vehicle drivers. If the studied parking lot is not regularly visited by the same vehicles, there is too little interest to attempt a vehicle [reID](#) with a constantly changing vehicle set. To give an example, it is much more pertinent to perform [reID](#) on a company staff parking lot rather than a motorway rest area parking where the probability to observe two times the same unique vehicle is greatly lower. In other cases, the algorithm should keep many tracks in order to find a possible match with incoming detections.

The algorithm performance will greatly depend on the cropping resolution. The cameras placement is crucial to obtain a full coverage of the parking lot but also to obtain multiple point-of-views of each vehicle with acceptable resolution. In addition, the elevation angle of the cameras should be well selected in order to both have an horizontal view of the vehicles to clearly perceive the salient details which are needed to perform vehicle [reID](#) and a sufficiently vertical view to avoid occlusions and increase the cameras coverage.

An efficient way to sometimes guarantee vehicle [reID](#) is to read the license plates and completely ignore the vehicles aspect. For example a well placed camera at the entrance of the parking lot can extract the license plate registration number and increment a counter for this vehicle in a small database. This kind of approach is capable to produce excellent results in particular situations where the line of sight to the license plate is always clear and does not suffer from occlusion. This is of course not always possible and researchers prefer to ignore license plates to produce an algorithm that can more easily generalize.

The [reID](#) algorithm presented in chapters [3](#) and [4](#) is designed to meet several specifications which have been discussed in this chapter. The algorithm:

- Does not use any license plate information.
- Does not handle night time and other strong light variation such as cloudy weather or rain.
- Uses and combines information from multiple cameras.
- Assumes that each vehicle is observed by at least one camera.
- Assumes that vehicle appearance does not change over time.

## Chapter 3

# Siamese network for vehicle reID

This chapter presents how the fundamental tool of the reID algorithm presented in chapter 4 is designed and trained. The performances of the proposed CNN will be discussed in chapter 5.

### 3.1 Approach and motivation

The application specified in chapter 2 aims to provide more visual diversity of the observed vehicles compared to other existing vehicle reID algorithms. The multiple observations of a single parked vehicle taken at a given instant will be compared with a database of past observations in order to find a match and perhaps re-identify the observed vehicle.

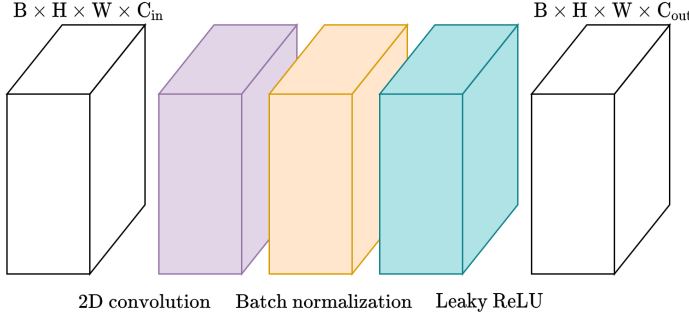
The presented network architecture is used by the multi-cameras vehicle reID algorithm presented in chapter 4. Since the algorithm is about comparing multi-views representations of vehicles for which the number can vary, the image comparison tool cannot know in advance how many viewpoints it will have to compare. Without this knowledge, the chosen approach is to create and train a siamese deep neural network able to remain consistent when comparing images taken by different cameras. The siamese network is trained using pairs of images as input and will be used the same way during inference, which solve the previously mentioned issue.

### 3.2 Architecture design

As discussed in section 1.4, a siamese network is trained on input pairs. Practically speaking, one does not necessarily have to duplicate a CNN architecture but rather could mimic the duplication principle by passing sequentially the images through the network and actuate the weights by comparing both outputs.

A good approach to design a CNN is to select a pretrained *backbone* and attach a task-specific *head* with it to avoid beginning from scratch. However the computational resources at disposition for the training phase were not able to hold the state-of-the-art *backbones* such as ResNet50 without causing an overload of the GPU RAM. This is the reason why a lighter *backbone* and *head* are built as the evaluation tool of the reID algorithm.





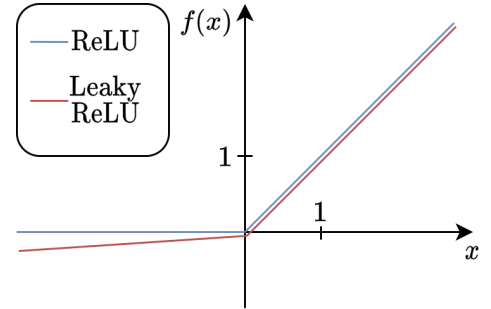
**Figure 3.1:** Structure of the CBLR element

The convention to treat the input or output dimensions in a CNN is to qualify them as objects of dimensions  $B \times H \times W \times C$  where:

- B is the batch size.
- H is the height.
- W is the width.
- C is the number of channels.

The fundamental element of the proposed CNN is the Convolution+Batch normalization+Leaky ReLU (CBLR) block. Its structure is presented on figure 3.1. This element is responsible to build and transform the feature maps at several levels in the CNN architecture. The 2D convolution layer is used to first transform the input by the means of local transformations, which are performed by the sliding kernel. The batch normalization layer helps the global network to train separate layers more independently by normalizing the input across the batches. It is also useful to prevent overfitting and has shown to provide an added value in the CNN training literature.

The Leaky ReLU layer is the activation function which introduces the non-linearity in the network. Leaky ReLU is a tweaked version of the classical ReLU activation function (figure 3.2) where its input-output relation is given by  $f(x) = \max(\alpha x, x)$  where  $\alpha$  describes the slope when  $x < 0$ . While ReLU activation solves the vanishing gradient issues during CNN training phase, it could erase the gradient when the input is negative and lead to locked network, where some branches of the network cannot be trained since the gradient is null. The leaky version is used to attempt to solve this potential issue by letting a small slope when the input is negative, resulting in a non-negative gradient. By doing so, the training phase is often more stable.



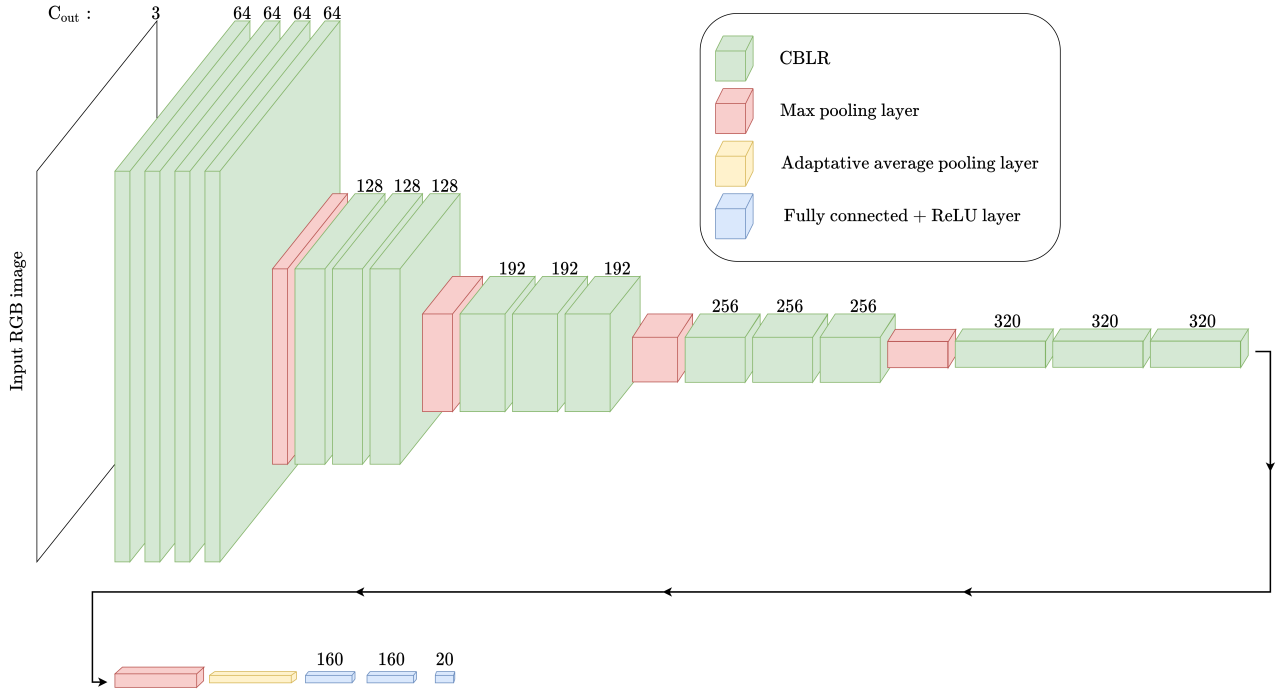
**Figure 3.2:** ReLU and Leaky ReLU activation functions

The global network architecture is presented on figure 3.3. The input is an RGB image or a pair of RGB images passed sequentially. Inspired by what has been proposed in [36] and by the structure of VGG-Net [17], the CNN is firstly composed of 4 CBLR blocks producing feature maps with 64 output channels. The feature maps are then downsampled using a maximum pooling layer in order to both preserve the main characteristics embedded in the feature maps and to increase the robustness to local variation in the previous feature maps.

The 64 channels are passed through 3 CBLR blocks which increase the number of channels to 128 before being downsampled by a maximum pooling layer. The process is repeated thrice to increase the number of channels to 192, 256 and 320 in this order. The *backbone* is completed by connecting an adaptive average pooling layer to reduce the feature maps to a single vector of size  $B \times 1 \times 1 \times 320$ .

The final feature map is input to a series of 3 fully-connected layers with each time a ReLU activation layer to decrease the features to sequentially 320, 160 and 20. This last  $B \times 20$  vector is the feature representation of the input image batch.

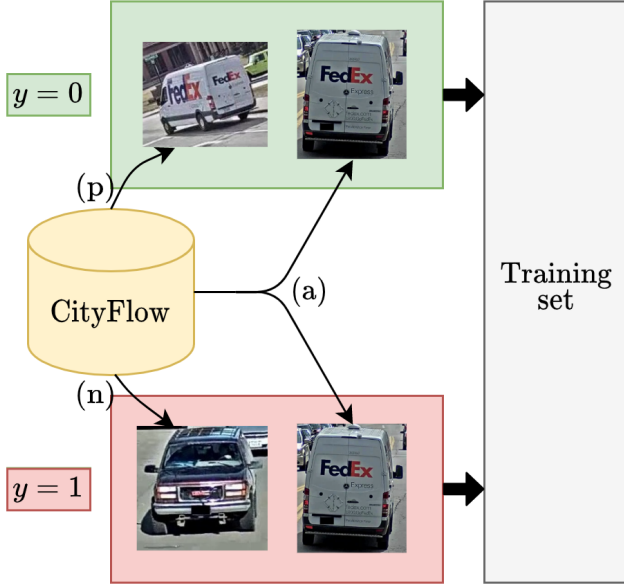
Concerning the non-trainable parameters, the slope  $\alpha$  of all Leaky ReLU activation functions was set to 0.15. Every convolutional layers within the CBLR blocks have a kernel size of  $3 \times 3$  with appropriate padding. All maximum pooling layers have a kernel size of  $3 \times 3$  and a stride (sliding step) of 2.



**Figure 3.3:** Deep neural network with channel representation

### 3.3 Network training

As discussed in chapter 2, the parking lot simulation does not provide enough diversity, as much in terms of vehicle models which are limited to 14 and as much in terms of context diversity where only one type of parking lot is simulated. The hand-crafted dataset cannot achieve this purpose either and will rather be used for the test phase in chapter 5.



**Figure 3.4:** Training set preparation

This is why a real dataset is preferred. To this end, the CityFlow[8] dataset is used for the training phase. It is important to feed the network with 50% of positive pairs and 50% of negative pairs in order to not induce any decision bias during the training step.

To do so, an anchor image is picked at random in the whole dataset. The positive image is selected at random among all the images with the same vehicle ID as well as the negative image among all the images carrying a different ID (figure 3.4). For an image pair  $\{X_1, X_2\}$ , each frame is passed sequentially through the whole neural network  $G_W(.)$  producing a pair of feature vectors:  $\{f_{X_1}, f_{X_2}\}$ .

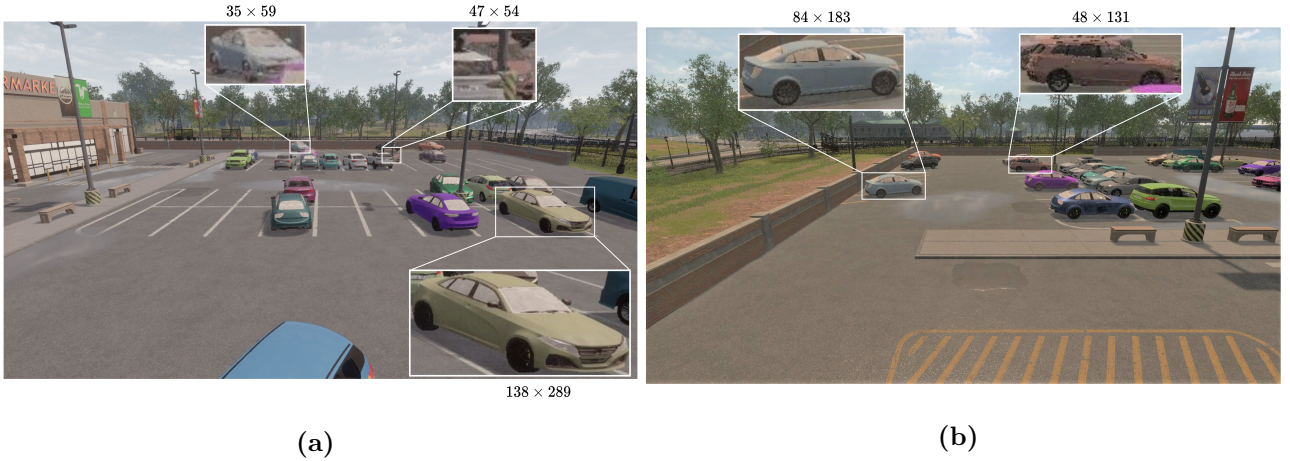
The contrastive loss defined at equation 1.3 is used to train the network. The margin value is arbitrary set to 4 to compare feature vectors of size 20, which is slightly higher than in the literature to avoid large deformation of the feature space. Thus, the CNN will tend to bring images from the same vehicle label as close as possible in the features space and repel different class clusters to a squared Euclidean distance of 4.

During the training phase, the batch size  $B$  was set to 16, the learning rate to 0.0001 and all the weights were initialized at random following a normal distribution  $\mathcal{N}(\mu = 0, \sigma^2 = 0.01)$ . The free GPUs of Google Colaboratory are used. It is trained during 10 epochs over 30,935 images of CityFlow which are depicting 333 different vehicles. The remaining 6000 images are kept as a test set. The performances of the trained siamese network are presented and discussed in chapter 5.

## Chapter 4

# Multi camera re-identification

Performing a vehicle re-identification task using a single camera could be challenging due to the lack of degrees of freedom in the vehicle representation. On figure 4.1a, one can observe an example where the vehicles parked in the background have poor resolution. In addition, cameras also suffers from occlusions and field of view limitation.



**Figure 4.1:** Two viewpoints of the same scene from the parking lot simulation

On the other hand, figure 4.1b shows another camera viewpoint of the same scene and offers a better resolution of the same vehicles that were previously in the background and partially occluded. Combining the different viewpoints can provide better chances to the reID algorithm of having at least one image with sufficient resolution and without any occlusion.

In the case of vehicle detection task, the low resolution is not so much an issue compared to the re-identification task, where the neural network will try to extract high and low-level semantic features in order to recognize the vehicle ID. The main idea presented in this chapter is to provide more information to the decision algorithm in charge of the re-identification and draw benefits from a scenario where a parking is observed by several cameras.



## 4.1 Multiple vehicle detections within a single viewpoint

Since the siamese network has been fed with cropped vehicle pairs during training, it is needed to extract efficiently each vehicle subframe that is present within a camera field of view. Therefore the first step of the multi-camera cooperation is to produce separate vehicle crops since the whole image cannot be input as it is into the siamese network without being pre-processed.

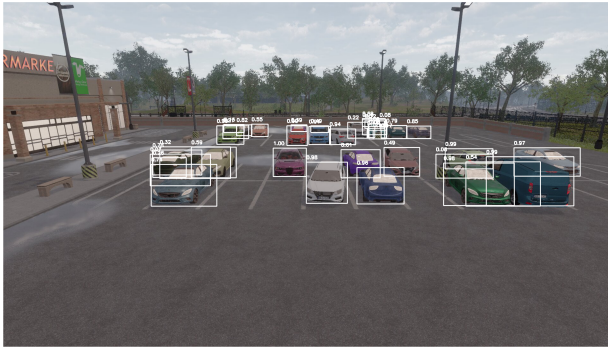


**Figure 4.2:** Example of probability mask extraction

The vehicles extraction step is performed using the object instance segmentation tool Mask R-CNN[42]. For a given input image, Mask R-CNN outputs a bounding box for each detection and also provides a probability mask (figure 4.2) and a confidence score for each class. In the present case only the cars are kept, other detections are ignored.

In addition, a Non Maximum Suppression[43] (NMS) method is applied over the raw detection output of Mask

R-CNN to deal with overlapping detection boxes. The improvement can be observed between figures 4.3a and 4.3b. NMS is useful to avoid redundant bounding boxes, but it can also suppress important detections resulting in undetected vehicles.



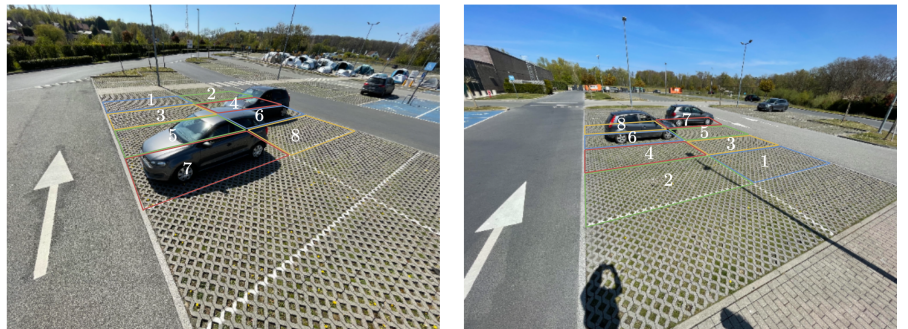
(a)



(b)

**Figure 4.3:** Comparison of vehicle detection with and without NMS

A practical question remains: once all global pictures have been cropped into individual vehicle images, how one could match the images of the same vehicle and group them? In the proposed reID algorithm, it is assumed that each camera field of view has been annotated with a segmentation of all the covered parking space

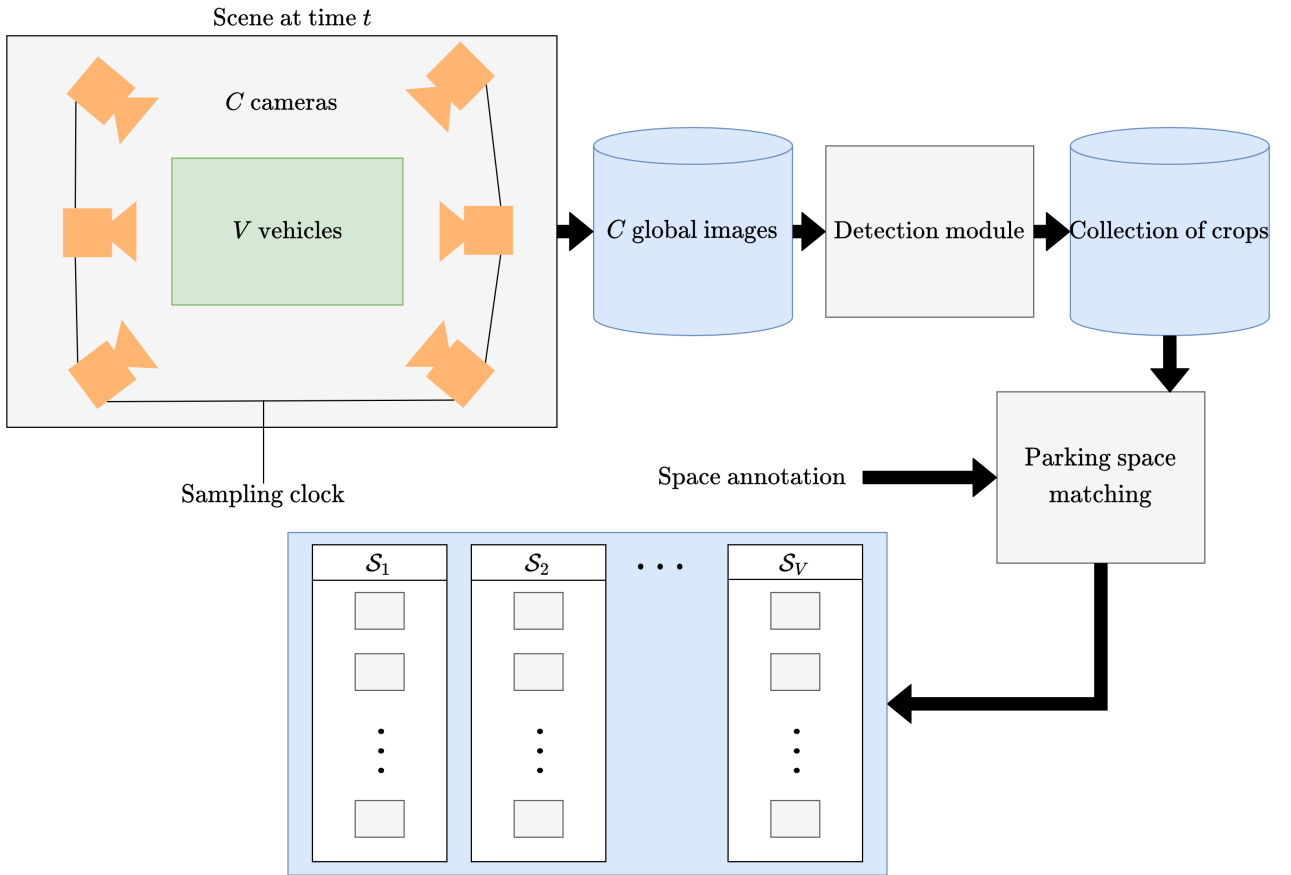


**Figure 4.4:** Parking space annotation

(figure 4.4). With such additional knowledge, the detections can be directly matched and grouped together with regard to the space annotation.

## 4.2 Stacking multiple vehicle representations

The state of the parking lot at a given time stamp is called a scene. The scene is composed of multiple vehicles placed on different parking spaces. The scene is sampled by  $C$  cameras at the same time. This is mandatory for the presented method: if a parking space is captured by several cameras and if a vehicle stands on that particular space, then it is guaranteed that the cameras will observe the same vehicle on this particular position. However, it does not mean that the line of sight on that particular parking space must be cleared for all the cameras since occlusions could happen.



**Figure 4.5:** Information processing for *shootings* construction

Figure 4.5 depicts the process for assembling efficiently the multiple views of the  $V$  vehicles within the scene. Once all the cameras have captured  $C$  global images of the observed scene, the detection module discussed in the previous section produces at most  $V \times C$  crops. Using an initial field of view segmentation for each camera, the cropping are gathered according to their parking space. The  $V$  detected vehicles are finally assigned several crops ranging from 1 to  $C$ . Each vehicle image stack is called a *shooting* and the  $i^{\text{th}}$  *shooting* is denoted by  $S_i$ . It can be interpreted as the multi-view representation of the concerned vehicle in the visual domain. With the collection of the  $V$  *shootings*, the *reID* algorithm will have to assign each *shooting* to a previous detected vehicle if possible.

What if the cameras were sampling the scene asynchronously ? In that case the probability of capturing several crops of the same vehicle drops, depending on how much the capture times are



misaligned. To give an example, one can imagine that a parking lot composed by a single parking space is observed by two cameras. In addition the parking space belongs to the field of view of both cameras and the line-of-sight is cleared so that no occlusion are envisaged. The cameras are sampling the scene every 20 minutes, but the second camera lagged the first one by a time shift of 10 minutes. In that case, this is possible that between the two sampling times, the observed vehicle leaves the parking space and is getting replaced by another vehicle. This particular situation raises a problem: stacking naively the two crops does not work due to the conflict of detected vehicle labels. The multi-view staking becomes more complicated since the field of view segmentation to match the parking spaces is not enough. To solve this issue, two solutions could be envisaged:

- One could simply ignore the mentioned issue and let the [reID](#) algorithm treats the *shooting* as it is in the hope that the most captured vehicle represented in the *shooting* is enough to successfully find a match. The other detections are therefore performing as parasite images which are degrading the multi-view representation of the main vehicle.
- Implement an additional vehicle verification step in the process. It consists of checking each detection within the *shooting* and attempts to separate the different vehicles. This solution is more complicated since it requires another vehicle [reID](#) step in charge of comparing single images in the shooting in order to apply a first weak discriminator and split the *shooting* according to the detected differences.

However if one assumes the cameras to be connected to the same computer (i.e., belong to the same network), the synchronous sampling requirement is not causing any important inconvenience.

### 4.3 Estimation of the quality of an image

When the [reID](#) algorithm attempts to compare image pairs, it is important to provide an *image score* for each vehicle crop in order to indicate whether or not the images the algorithm is analyzing are relevant for a truthful comparison. If taken into consideration, such quality estimation metric will indicate to the algorithm how to select pertinently the information in the diversity offered by a multi-view representation of a vehicle.

In this section, a mathematical computation which aims to estimate the quality of an image for efficient vehicle [reID](#) is proposed. To build this quality metric, one should think about the elements that make an image well suited for vehicle [reID](#) problem and verifies if the metric improve the final results. Ideally an image with great quality should:

1. Have a sufficiently large resolution.
2. Have a correct height/wide ratio. Since Mask R-CNN can also detect vehicles located at the edge of the field of view or detect a part of a vehicle which is partially occluded, one prefers to trust crops that are close to a square shape.
3. Not suffer from occlusions.
4. Represent a vehicle.
5. Be captured in the region of interest of the camera.
6. Be captured by a camera with a pertinent and useful point-of-view of the scene.
7. Be captured with ideal illumination conditions.

Specifications 1 and 2 can be handled in a common metric. The *resolution score* for an image of size  $H \times W$  (ignoring the color channels in the notation) is defined as follow:

$$\text{resolution score} = \min\left(1, \min(H, W)/224\right) \in [0, 1] \quad (4.1)$$

That is, only the smallest dimension is taken into account and divided by 224. It is divided by 224 since the deep neural network has been trained with images of size  $224 \times 224$  and therefore images with higher resolution do not bring any additional information in this case. Another approach is to compute the number of pixels  $H \cdot W$  contained in the image. However computing the area does not indicate if the crop is a thin rectangle. Taking the minimum instead punishes rectangles with the smaller dimension inferior to 224 pixels.

Specification 3 is more complex to handle if one only has access to the images without additional information. If the cameras can also provide a depth map, it is possible to verify that the detection matches the expected depth of a particular parking space. If the vehicle does not match the expected depth map, it is more likely that the detection is in fact an occlusion in the line of sight of the camera observing that particular parking space. To keep it general, one could define an *occlusion score* with its definition depending on the particular application and equipment.

Specification 4 can be handled by looking at the confidence score of Mask R-CNN which is providing *confidence scores* belonging in  $[0, 1]$ .

Specification 5 is application dependent. The user may indicate to each camera on which part of their field of view they have to focus on by using a mask. For example, if a road in the scene has to be ignored by the detection algorithm, it can be decided by observing whether or not the detection belongs to the desired field of view of the camera.

Specification 6 follows the idea that a vehicle is easier to re-identify if it is possible to observe the salient element of its shape, such as the windows, the hubcaps, the roof or the headlights. In order to catch a maximum of vehicle shape information, a horizontal three-quarter view is often the best. But as discussed in chapter 2, horizontal view leads to more occlusions and thus less detections. If a camera has a top-view of the scene, it will not suffer occlusions but it will struggle to distinguish two different vehicle shapes. A compromise is to place the camera with a three-quarter diving view of  $45^\circ$  in order to have a trade-off between occlusions and vehicle information. However, since this is not always possible, the user can provide a *view score* between  $[0, 1]$  for each camera describing its view quality.

Specification 7 implies a particular attention for scenario with important light variations. The illumination quality is difficult to quantify since it depends on the time of day, the weather, the reflections, the shadows, the signal to noise ratio value and so on. A basic solution is to attach an image with an *illumination score* in  $[0, 1]$  which depends on the light perceived by a daylight sensor.

Assembling all the specifications together, the *image score* is defined as follow:

$$\text{image score} = \text{resolution score} \times \text{occlusion score} \times \text{confidence score} \times \text{view score} \times \text{illumination score} \quad (4.2)$$

and belongs in  $[0, 1]$ . If a detection is out of the ROI, it is simply ignored. The choice of successive multiplication of all the discussed scores pushes the algorithm to rely more on images which are matching all the specifications at once. Indeed an image should meet all the specification to get a score close to 1 since any missing specification could drive its *image score* close to 0.

#### 4.4 Database design

It is important to design an efficient structure to store previous detections. Each time a *shooting* is received, the *reID* algorithm will attempt to match it with another multi-view representation of a vehicle contained in the database.

Figure 4.6 depicts this structure. The database in charge of recording past detections is composed of a limited amount of memory chunks where each chunk can store a particular multi-view representation. In term of naming, such a memory chunk is called a *gallery* since it contains several representations of the same vehicle. The  $i^{\text{th}}$  *gallery* is denoted by  $\mathcal{G}_i$ . All the galleries are gathered into a *museum* denoted by  $\mathbf{G}$ . The number of *galleries* is limited to  $|\mathbf{G}|$  and each *gallery* can store a maximum of  $R^{\max}$  different representations. In addition every *gallery* stores the *image score* of each representation and also keeps a track list recording all the time stamps for which the represented vehicle was detected.

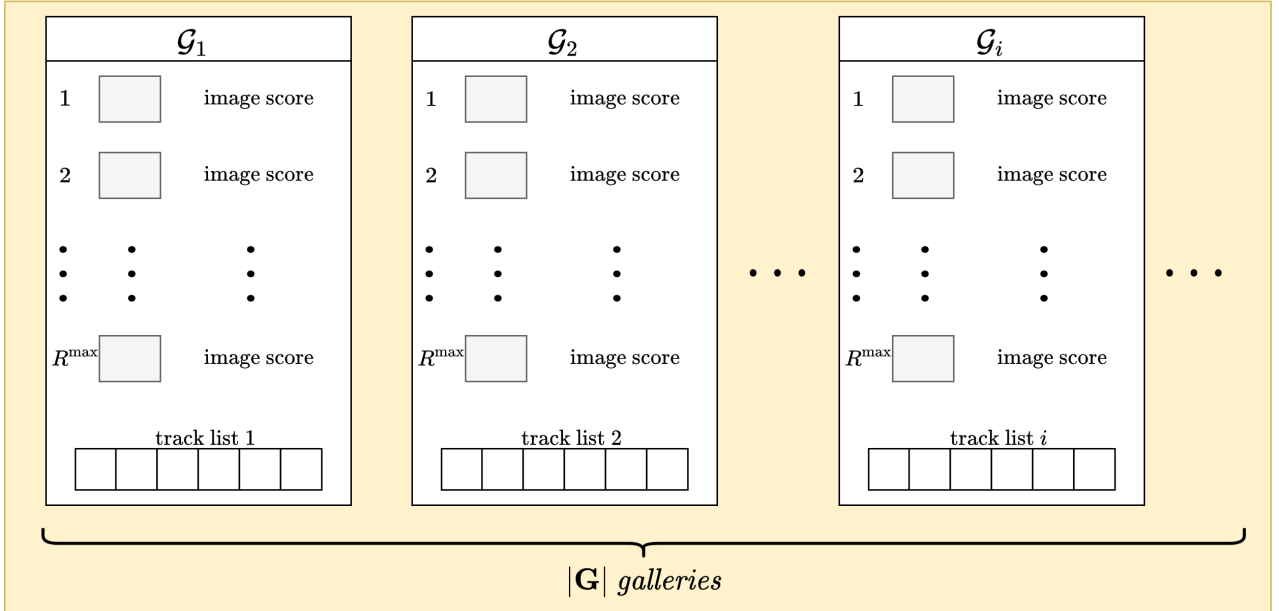


Figure 4.6: Structure of the *museum* database  $\mathbf{G}$

This database will be updated each time the camera network is sampling the observed parking lot. The matching algorithm and the database update are explained in section 4.7.

#### 4.5 Comparing two multi-view representations

In this section another metric is proposed to quantify how much a multi-view detection captured by the cameras network matches a given multi-view representation stored in the database. Without loss of generality, the *shooting*  $\mathcal{S}_j$  is composed of  $|\mathcal{S}_j|$  images and *gallery*  $\mathcal{G}_i$  contains  $|\mathcal{G}_i|$  representations of a single vehicle.

Based on the siamese network output presented in chapter 3, the *pair score* between the  $k^{\text{th}}$  image of  $\mathcal{G}_i$  and the  $l^{\text{th}}$  image of  $\mathcal{S}_j$  is:

$$s_{kl} = D_W(X_k^{\mathcal{G}_i}, X_l^{\mathcal{S}_j}) \quad (4.3)$$

Where  $D_W(\cdot, \cdot)$  is the euclidean distance defined at equation 1.2.

Based on the *pair score* metric, the similarities between *gallery*  $\mathcal{G}_i$  and *shooting*  $\mathcal{S}_j$  are captured by the *matching matrix*  $\mathcal{M}_{ij}$  which is defined as follows:

$$\mathcal{M}_{ij} = \begin{bmatrix} s_{11} & \cdots & s_{1l} & \cdots & s_{1|\mathcal{S}_j|} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ s_{k1} & \cdots & s_{kl} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ s_{|\mathcal{G}_i|1} & \cdots & \cdots & \cdots & s_{|\mathcal{G}_i||\mathcal{S}_j|} \end{bmatrix} \quad (4.4)$$

This matrix contains  $|\mathcal{G}_i| \times |\mathcal{S}_j|$  positive entries, each one being the comparison of two feature representations. Entries close to 0 indicate that their corresponding images pair are likely from the same vehicle ID.

In order to decide whether or not the information agglomerated in the *matching matrix*  $\mathcal{M}_{ij}$  can be trusted, the *image score* metric is used to decide how much an entry  $s_{kl}$  has been computed with ideal conditions regarding the specifications mentioned in section 4.3. Equation 4.5 dictates how much the value of  $s_{kl}$  should be trusted:

$$\xi_{kl} = \min\left(\text{image score}(X_k^{\mathcal{G}_i}), \text{image score}(X_l^{\mathcal{S}_j})\right) \in [0, 1] \quad (4.5)$$

That is, the value of  $\xi_{kl}$  between two images indicates the information quality contained in  $s_{kl}$  and will be close to 1 if both corresponding images have an *image score* close to 1. The  $\min(.,.)$  operator is used to pay more importance to image pair with two high *image scores*.

Finally, the *attribution cost*  $\mathcal{C}_{ij}$  between the *gallery*  $\mathcal{G}_i$  and the *shooting*  $\mathcal{S}_j$  is computed as the mathematical mean of all the *matching matrix*  $\mathcal{M}_{ij}$  entries weighted by the coefficients defined at equation 4.5:

$$\mathcal{C}_{ij} = \frac{1}{|\mathcal{G}_i||\mathcal{S}_j|} \frac{\sum_{k=1}^{|\mathcal{G}_i|} \sum_{l=1}^{|\mathcal{S}_j|} \xi_{kl} s_{kl}}{\sum_{k=1}^{|\mathcal{G}_i|} \sum_{l=1}^{|\mathcal{S}_j|} \xi_{kl}} \quad (4.6)$$

Thanks to this quantity, the **reID** algorithm presented in section 4.7 will be able to quantify information.

## 4.6 Assignment to the database

Once the *attribution costs* are computed, one has to decide what to do with these quantities to decide how to assign the *shootings* representing the detected vehicles to the *galleries* contained in the database.

One way to do so is to assign sequentially each *shooting*  $\mathcal{S}_j$  with the *gallery*  $\mathcal{G}_i$  such that the corresponding *attribution cost*  $\mathcal{C}_{ij}$  is minimum in the set  $\{\mathcal{C}_{kj}\}_{k=1}^{|\mathcal{G}|}$ . However this greedy approach is not the best since it could lead to suboptimal assignment.

The Hungarian method[44] is a optimization algorithm that solves the assignment problem in polynomial time. In other words, there is an optimal way to assign  $N$  workers to  $N < M$  distinct jobs (figure 4.7) such that the total cost of the final assignments is minimum.

|                           |  | Workers ( <i>shootings</i> ) |          |                    |          |                    |
|---------------------------|--|------------------------------|----------|--------------------|----------|--------------------|
| Jobs ( <i>galleries</i> ) |  | $\mathcal{C}_{11}$           | $\cdots$ | $\mathcal{C}_{1j}$ | $\cdots$ | $\mathcal{C}_{1N}$ |
|                           |  | $\vdots$                     | $\ddots$ | $\ddots$           | $\ddots$ | $\vdots$           |
|                           |  | $\mathcal{C}_{i1}$           | $\ddots$ | $\mathcal{C}_{ij}$ | $\ddots$ | $\mathcal{C}_{iN}$ |
|                           |  | $\vdots$                     | $\ddots$ | $\ddots$           | $\ddots$ | $\vdots$           |
|                           |  | $\mathcal{C}_{M1}$           | $\cdots$ | $\mathcal{C}_{Mj}$ | $\cdots$ | $\mathcal{C}_{MN}$ |
| No match                  |  | $\mathcal{T}$                | $\cdots$ | $\mathcal{T}$      | $\cdots$ | $\mathcal{T}$      |
|                           |  | $\vdots$                     | $\ddots$ | $\ddots$           | $\ddots$ | $\vdots$           |
|                           |  | $\mathcal{T}$                | $\cdots$ | $\mathcal{T}$      | $\cdots$ | $\mathcal{T}$      |

**Figure 4.7:** Cost matrix

Let  $\mathbf{A}$  be the assignment matrix for which  $\mathbf{A}_{ij} = 1$  if worker  $i$  is assigned to job  $j$  and 0 otherwise and  $\mathbf{C}$  be the cost matrix. In that case, the Hungarian method find the optimal solution to the following minimization problem written at equation 4.7:

$$\mathbf{A}^{\text{opt}} = \arg \min_{\mathbf{A}} \sum_{i,j} \mathbf{A}_{ij} \mathbf{C}_{ij} \quad (4.7)$$

To make the link with the application, the *shootings* are the workers that one would like to assign to the *galleries* which are the jobs. In addition a collection of fake jobs are created to handle the case when a detected vehicle does not find any relevant match in the database. For these particular scenarios, the cost matrix is extended with a sufficient number of fake jobs for which the assignment means the creation of a new *gallery* in the database. These new jobs have an *attribution cost* of  $\mathcal{T}$  with a threshold value depending on the scenario.

## 4.7 Full multi-camera vehicle reID algorithm

In the last part of this chapter, the complete algorithm is presented. This vehicle reID algorithm combines all the discussed aspects of this chapter and can be separated in two parts: the initialization of the database at time  $t = 0$  and a typical iteration step when  $t > 0$ . At each time step, this algorithm is responsible to efficiently update its database to keep track of detected vehicles and manage the available resources.

### 4.7.1 Database initialization

Figure 4.8 depicts the diagram for the initialization step of the algorithm at time  $t = 0$ . The database  $\mathbf{G}$  is an empty structure and therefore does not contain any vehicle representation. Depending on the user needs and available memory space, the maximum amount of *galleries*  $|\mathbf{G}|$  and the maximum number of representations per *gallery*  $R^{\text{max}}$  must be specified for the whole reID process.

The parking space is first sampled by the cameras network to produce  $C$  global images. From this point, the *shootings* are constructed by first detecting a maximum of vehicle crops among the  $C$  global images after what the crops are gathered in multi-view representation stacks according to the space annotation (discussed in section 4.2).

The *image score* of each image contained in the  $V$  *shootings* is computed in order to attach it with its quality estimation responsible to penalize the crops that do not meet the specifications of a typical image of good quality (section 4.3).

Finally the  $V$  *shootings* representing  $V$  distinct vehicles are assigned to  $V$  *galleries* in the database  $\mathbf{G}$ . If for a particular *shooting* the number of images is superior to  $R^{\text{max}}$ , only the  $R^{\text{max}}$  images with the greater *image scores* are kept and stored in a *gallery*. The  $V$  updated *galleries* also update their track lists to mention the time stamp of the detection.

In case of insufficient space in the database to welcome the  $V$  *shootings*, the *shootings* are sorted first according to their number of images and secondly according to their average *image score*. Only the  $V$  first *shootings* are therefore stored in  $\mathbf{G}$ . If this kind of situation happens at the initialization, one should reconsider the memory resources allocated to the database since it might be too small to efficiently track the diversity of vehicles present in the scenario.

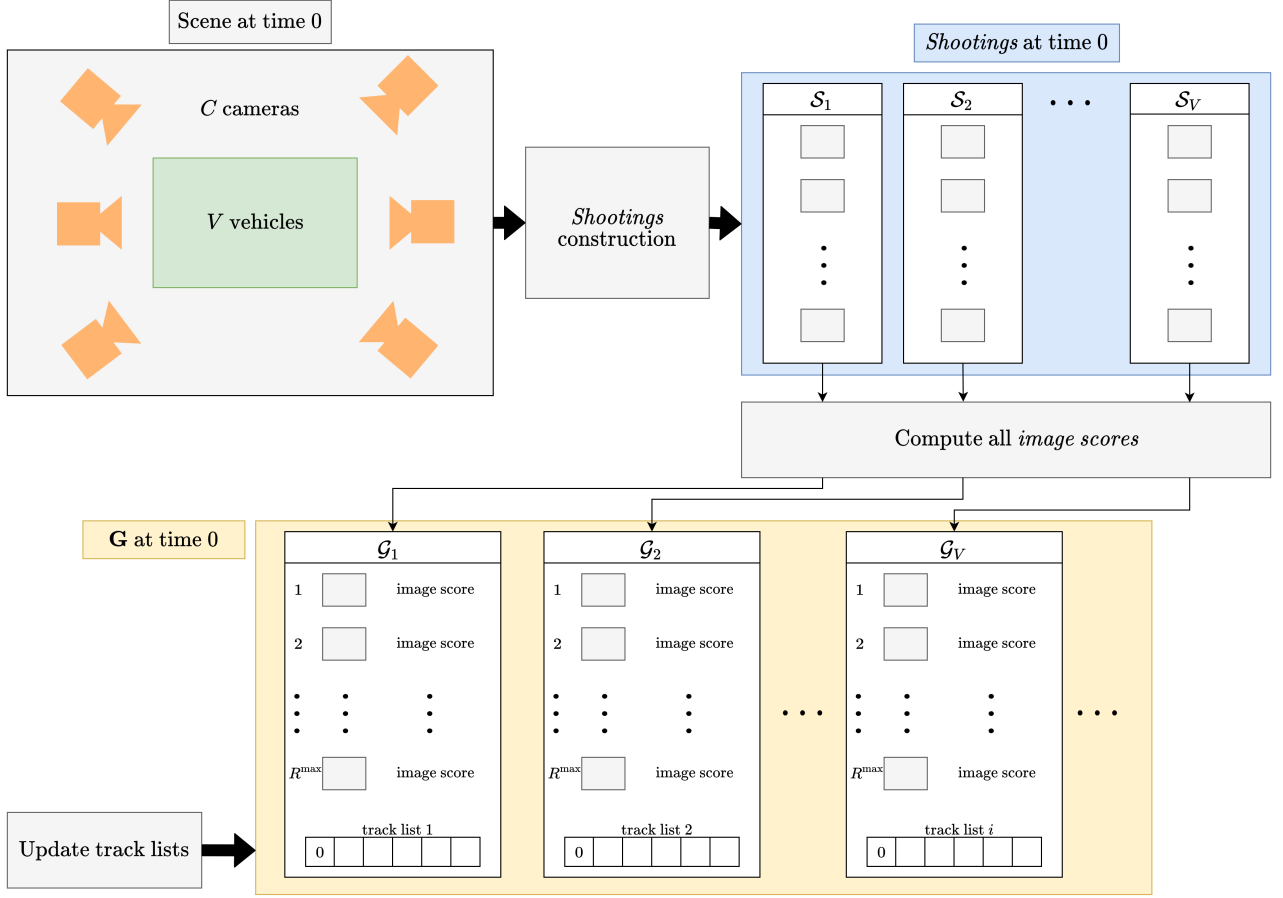


Figure 4.8: Initialization phase

#### 4.7.2 Iteration phase

Figure 4.9 depicts the diagram for a typical iteration step of the algorithm when  $t > 0$ . The database  $\mathbf{G}$  at time  $t - 1$  contains a certain amount of occupied and empty *galleries* with their respective multi-view representation, the *image scores* and the corresponding track lists.

As for the initialization phase, the scene at time  $t$  is sampled by the cameras network to produce global images which are used to construct  $V$  *shootings* representing the  $V$  vehicles present at time  $t$  on the parking lot. In addition all crops receive their *image scores*.

Once it is done, all possible pairs of *shootings* and occupied *galleries* are used to build their corresponding *matching matrices*. Each *matching matrix* compares the similarities between a multi-view representation in a *gallery* with a multi-view representation contained in a *shooting*. Furthermore each entries of a *matching matrix* is the Euclidean distance (equation 4.3) between two vehicle representations in the feature space produced by the deep neural network (discussed in chapter 3 and in section 4.5).

The collection of weighting coefficient  $\{\xi_{kl}\}$  (defined at equation 4.5) is computed for each *matching matrix* by combining the *image scores* of the corresponding *gallery* and *shooting*.

Every *matching matrices* are transformed into *attribution costs* by averaging them with the previously mentioned weighting coefficients (equation 4.6) in order to produce the cost matrix that will be



needed for the assignment to the database.

The cost matrix is extended with fake jobs of a cost value of  $\mathcal{T}$  as discussed in section 4.6 before performing an optimal assignment using the Hungarian method.

The *shootings* that are matching a particular occupied *gallery* see their images and corresponding scores being incorporated in the *gallery* collection if it is not completely full. Otherwise the *gallery* keeps only the  $R^{\max}$  images with the greater *image score* among the stored images and the *shooting*.

For those who are assigned to a fake job, they fill an empty *gallery* to create a new vehicle track if possible. If no empty *galleries* are left, they replace the oldest *galleries* if they have a larger amount of images and a higher mean *image score* in case of equality. It means that the previous vehicle track is deleted due to insufficient memory in the database.

Finally the track lists are updated by adding the timestamp  $t$  or by refreshing the track list in case of deleted track.

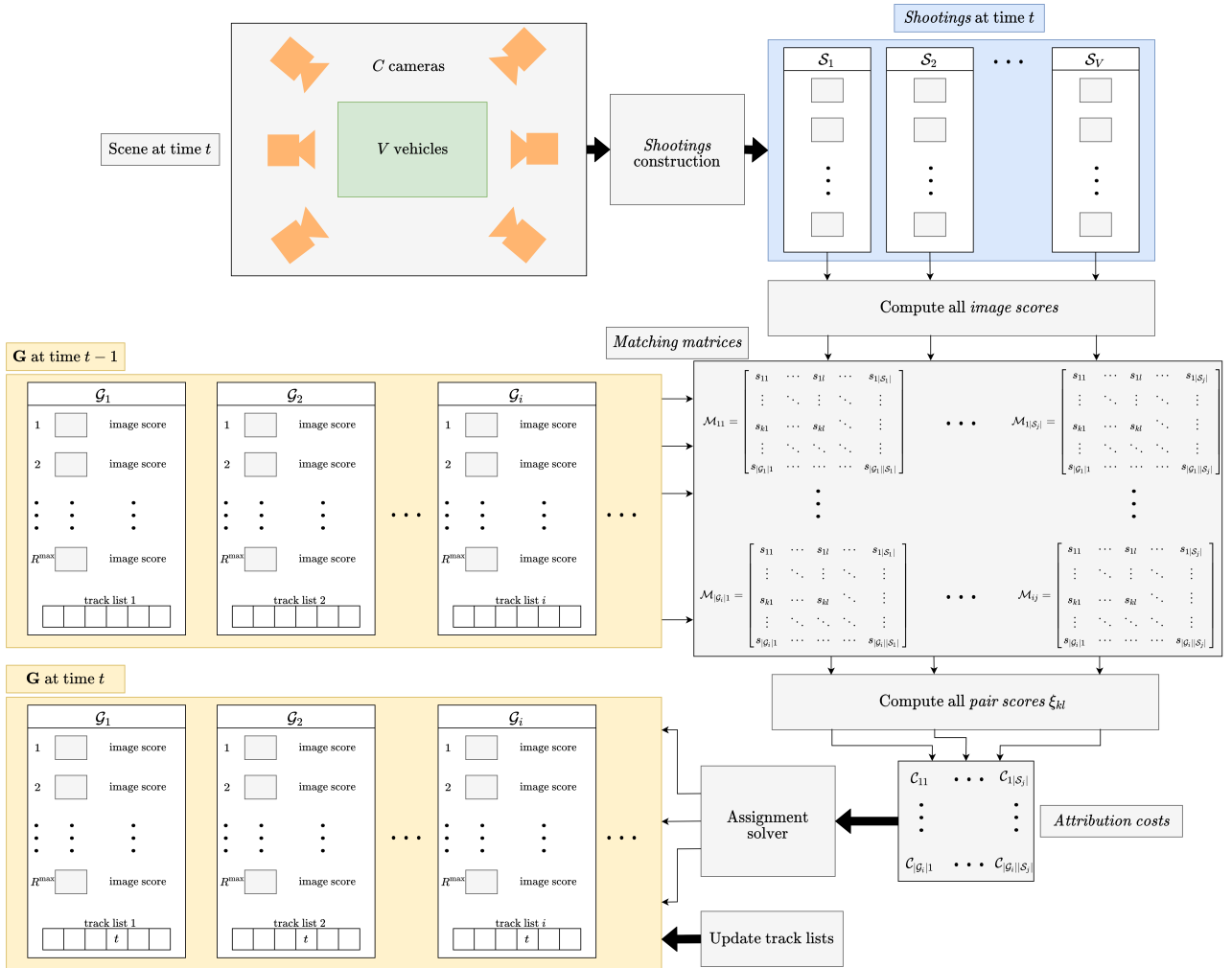


Figure 4.9: Iteration phase

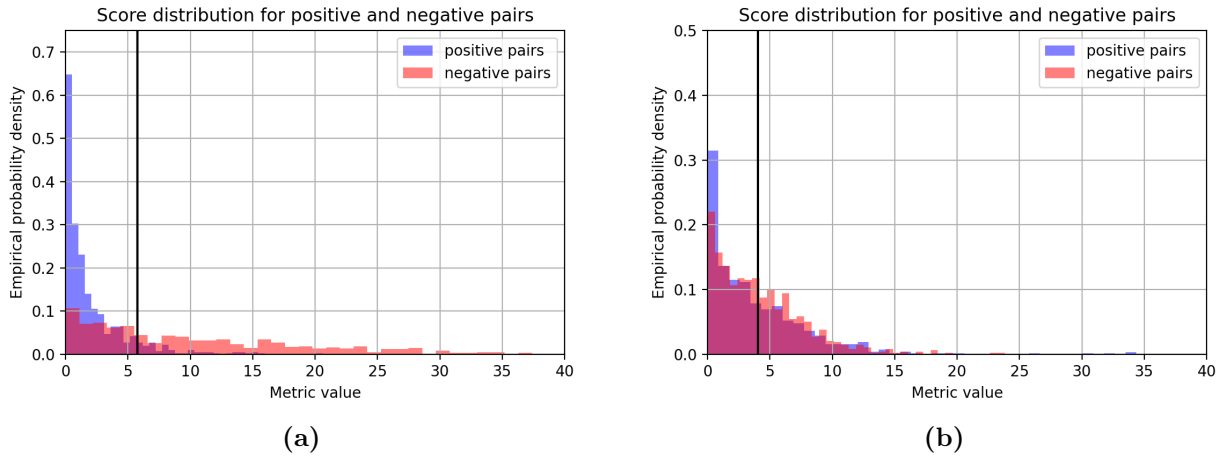
The presented algorithm is tested and discussed in chapter 5.

## Chapter 5

# Presentation of results

### 5.1 Siamese network performance

To investigate the performances of the CNN presented in chapter 3, 6000 images of CityFlow dataset have been kept as a test set. 3200 image pairs are assembled with a proportion of 50% of positive pairs and 50% of negative pairs. Each pair is forwarded across the trained network to produce a pair of feature vectors. The Euclidean distance (equation 1.2) between the two vectors is computed and registered.



**Figure 5.1:** Siamese network performances over 5.1a CityFlow and 5.1b hand-crafted test sets

Figure 5.1a shows the distribution of the distance between two image representations in the feature space. As one could expect, the trained network has brought images under the same label closer in the feature space and has repealed images with different labels. Based on the collected metric values, a binary classifier has been trained to find the optimal decision threshold (black line on the graph). For this model, the threshold value is 5.773 and it classifies correctly 76.06% of the pairs.

This siamese model was also tested over the hand-crafted dataset to estimate how it performs with a parking lot set. Figure 5.1b shows the metric value distribution. With an optimal decision threshold value of 4.033, it classifies correctly 53.5% of the pairs. As one can observe, the trained model is not able to split and dissociate the positive and negative pairs distributions. The poor classification score of 53.5% indicates that the model is just slightly better than a random guessing.

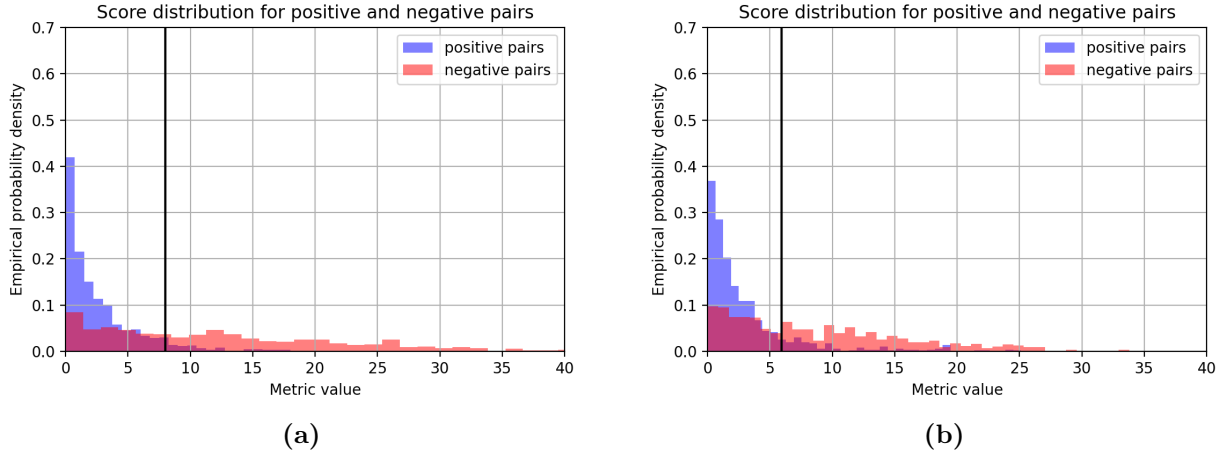
This issue is probably due to the fact that the handcrafted dataset contains only three different

vehicles with similar shapes and colors as discussed in chapter 2. It can also come from the difference of context: CityFlow is composed of vehicles on roads while the handcrafted dataset contains vehicle on a parking lot, leading to different kind of occlusions.

This issue preventing the test for the multi-camera reID algorithm, it was decided to train again the model on images pairs combining in each batch 80% and 20% pairs coming from respectively CityFlow and the hand-crafted dataset to mimic a model that can perform on parking lot data even if the diversity is not enough.

Since the handcrafted dataset is used both for training and testing, this second training phase was interrupted earlier to obtain a classification score on the parking lot dataset close to the one obtained on CityFlow and thus imitate the performance of a non-overfitting model. This choice induces that the re-trained model cannot be used for other parking lot scenario. While it is limiting the generalization capabilities of the network, it still allows to test the reID algorithm and discuss the importance of each parameter.

Therefore another model was trained using at initialization the weights of the previous one. The batch size was set to 8+2 by merging both datasets and the training was stopped after one epoch over the whole CityFlow training set. The obtained model was tested on CityFlow and the hand-crafted dataset as previously.



**Figure 5.2:** Re-trained network over 5.1a CityFlow and 5.1b hand-crafted test sets

The result over CityFlow is shown on figure 5.2a. With an optimal decision threshold of 8.021, it reaches a score of 75.07% of the pairs correctly classified. This result indicates that the re-trained model did not lose too much in term of performances over its original training set.

The output metric distribution with hand-crafted dataset is shown on figure 5.2b. It correctly classifies 72.22% of the pairs with a threshold value of 5.939. When comparing to figure 5.1b, one can observe that the new model is able to distinguish the two distributions and separate them in the feature space. Thanks to this model it is possible to perform several experiments on the multi-camera vehicle reID algorithm. However it is important to remain careful since the model was trained to differentiate only 3 vehicles and is not representative of the more complex diversity present in a typical parking lot scenario.

## 5.2 Multi-camera reID algorithm performance

After designing the multi-camera vehicle reID algorithm, it is important to qualify its performance at several levels in order to interpret its mechanisms and discuss the impact of its composing parameters. The two main questions that should be asked are:

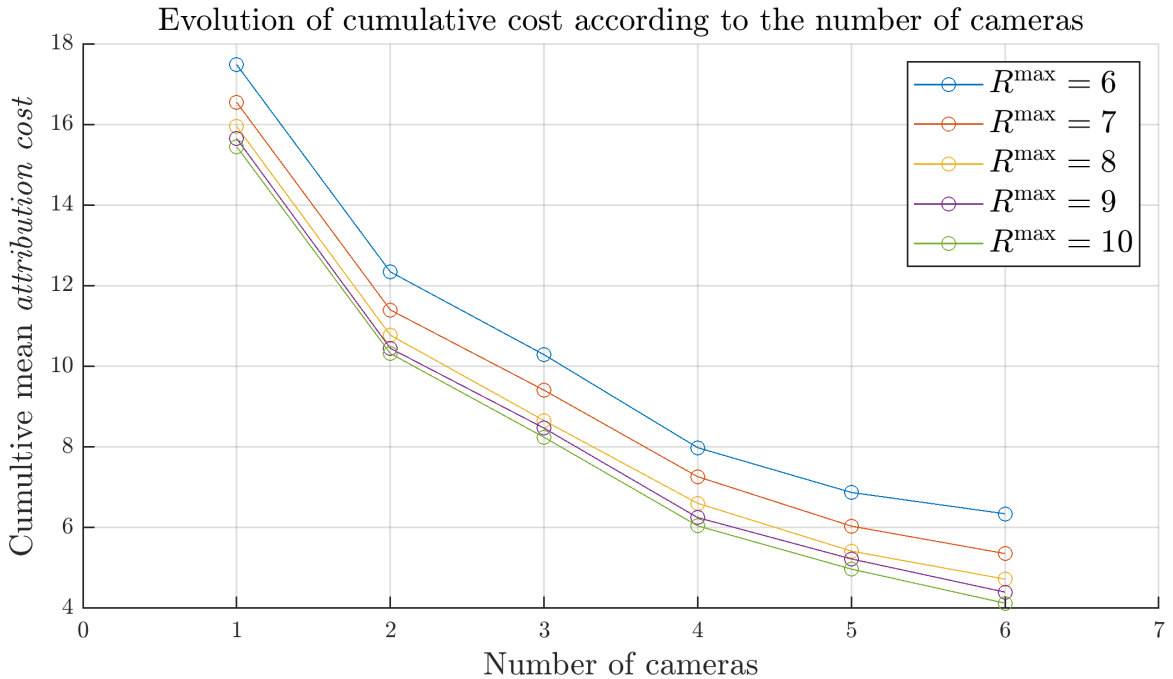
1. How does the algorithm assignment process behaves while varying the number of available cameras and the database size ? What is the *image score* impact on the assignment performance ?
2. How well does the algorithm is able to track the vehicles and how does the number of cameras, the number of spaces in the database, and the *image score* metric influence the results ?

To answer the first question, it is not possible to use the mAP metric since this is not an instance retrieval in a query problem as discussed in section 1.3 but rather a multiple vehicles tracking problem.

The chosen approach consists in looking at the accumulation of all the *attribution costs* (equation 4.6) of the algorithm on the 10 scenes of the handcrafted dataset. The assignment to the database is performed with the Hungarian method (section 4.6) which is minimizing the *attribution costs* at time  $t$ . If the algorithm performs poorly, it may assign each detection to a new *gallery* in the database which is represented by an *attribution cost* of value  $\mathcal{T}$ . For the experimentation,  $\mathcal{T}$  is set to 5.989 which corresponds to the optimal threshold value for the classification of image pairs from the handcrafted dataset (discussed in the previous section 5.1).

For the experimentation, the *image score* is only dictated by the *resolution score* (equation 4.1) since other parameters cannot be quantified with the handcrafted dataset. However, it helps in analyzing the impact of image resolution on the algorithm result.

In a first experiment, the algorithm is applied with different numbers of available cameras and different *gallery* sizes  $R^{\max}$ . The cumulative *attribution cost* is averaged over all possible combinations of cameras ( $\binom{n}{6}$  combinations for  $n$  cameras out of 6).



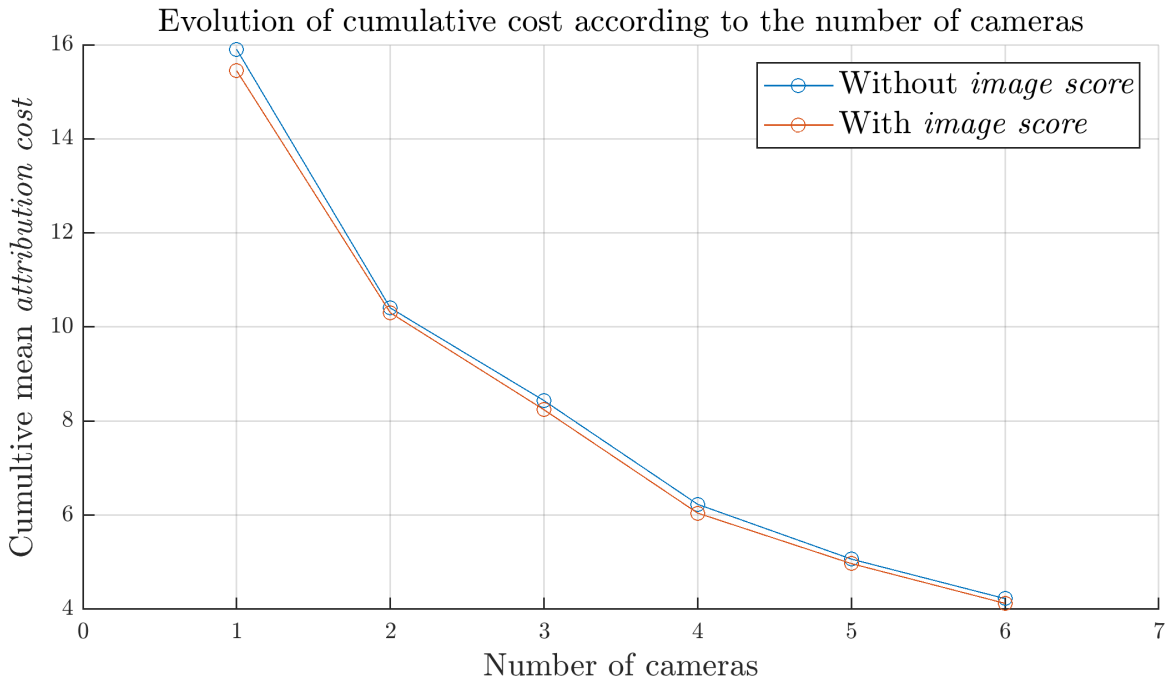
**Figure 5.3:** Algorithm experimentation 1

Figure 5.3 shows the mean cumulative *attribution cost* in function of the number of cameras for different values of  $R^{\max}$ . The algorithm performs better assignment choices when the number of cameras increases, proving the interest of using several cameras to make a decision when comparing two vehicles.

According to the curves shape, the performances may reach a plateau if the number of cameras continues to increase. It is intuitive: the number of possible viewpoints that can bring different pieces of information is limited, the visual diversity is not an unlimited source of information therefore an important number of cameras may not be a judicious choice since it will just slow down the algorithm computations and use resources in an unnecessary manner.

The number of maximum representations  $R^{\max}$  recorded in a *gallery* also has a positive impact on the algorithm performance. It also seems that increasing this parameter can improve the performance up to a certain point where it reaches a plateau. Using the same argument as for the number of used cameras, it may be useless to store too many vehicle representations since the visual aspect of a vehicle cannot bring unlimited information.

However these curves indicate that the user can obtain the same re-identification performances for large number of cameras but small  $R^{\max}$  and small number of cameras but large  $R^{\max}$ . For example the cumulative mean *attribution cost* for parameters  $\{C = 4, R^{\max} = 8\}$  is similar to the configuration  $\{C = 6, R^{\max} = 6\}$ . Therefore the user, depending on the situation, can decrease the number of cameras and increase the allocated memory in the database to store more vehicle representations per *gallery* which is often leading to a cheaper implementation.



**Figure 5.4:** Algorithm experimentation 2

Figure 5.4 shows a comparison of the algorithm assignment performance with and without taking into account the *image score* metric when averaging the *matching matrices* (equation 4.6), the parameter  $R^{\max}$  is set to 10. Without the consideration of *image score*, the algorithm treats every pair comparisons with the same level of confidence without favoring images with better resolution.

As one could observe, the consideration of *image score* for the selection of information only brings a

small improvement on the cumulative mean *attribution cost*. This behavior does not seem to depend on the number of used cameras or  $R^{\max}$ . However the interest of using the *image score* for information selection is better shown when looking at the final vehicle track lists.

To answer the second question, the chosen approach is to observe the output track list of each *gallery* after the algorithm is applied to the 10 scenes. To better understand the importance of each parameter, only eight configurations are discussed:

1.  $C = 3$  cameras,  $R^{\max} = 6$  representations per *galleries*, *image score* not considered.
2.  $C = 3$  cameras,  $R^{\max} = 10$  representations per *galleries*, *image score* not considered.
3.  $C = 3$  cameras,  $R^{\max} = 6$  representations per *galleries*, *image score* considered.
4.  $C = 3$  cameras,  $R^{\max} = 10$  representations per *galleries*, *image score* considered.
5.  $C = 6$  cameras,  $R^{\max} = 6$  representations per *galleries*, *image score* not considered.
6.  $C = 6$  cameras,  $R^{\max} = 10$  representations per *galleries*, *image score* not considered.
7.  $C = 6$  cameras,  $R^{\max} = 6$  representations per *galleries*, *image score* considered.
8.  $C = 6$  cameras,  $R^{\max} = 10$  representations per *galleries*, *image score* considered.

Other configurations behave in a similar way and the comparison of these 8 cases is therefore sufficiently representative of the algorithm's behavior.

The table 5.1 shows the output track lists for each configuration in the same order as above. The three vehicles are differentiated with distinct colors. The second row contains the true track list of each vehicle which indicate the time stamp of apparition on the parking lot. For example, the vehicle labeled by ID:2 is present on the parking lot at time stamps 0,1,2,5,6,7 and 9.

|         |                 | ID:1                     | ID:2                  | ID:3                  | Other track lists |        |
|---------|-----------------|--------------------------|-----------------------|-----------------------|-------------------|--------|
|         |                 | {0, 2, 3, 4, 6, 7, 8, 9} | {0, 1, 2, 5, 6, 7, 9} | {0, 1, 3, 5, 6, 8, 9} |                   |        |
| $C = 3$ | $R^{\max} = 6$  | {0, 3, 4, 7, 8, 9}       | {0, 2, 3, 5, 6, 8, 9} | {0, 2, 1, 6, 9}       | {1, 6}            | {5, 7} |
|         | $R^{\max} = 10$ | {0, 2, 3, 4, 7, 8, 9}    | {0, 2, 3, 5, 6, 8, 9} | {0, 1, 5, 6, 9}       | {1, 6}            | {7}    |
| $C = 3$ | $R^{\max} = 6$  | {0, 2, 3, 4, 6, 7, 8, 9} | {0, 1, 6}             | {0, 1, 5, 6, 9}       | {2, 5, 7, 9}      | {3, 8} |
|         | $R^{\max} = 10$ | {0, 2, 3, 4, 6, 7, 8, 9} | {0, 1, 6}             | {0, 1, 5, 6, 8, 9}    | {2, 5, 7, 9}      | {3}    |
| $C = 6$ | $R^{\max} = 6$  | {0, 2, 3, 4, 7, 8, 9}    | {0, 2, 5, 6, 7, 9}    | {0, 1, 5, 6, 8, 9}    | {1, 6}            | {3}    |
|         | $R^{\max} = 10$ | {0, 2, 3, 4, 7, 8, 9}    | {0, 2, 5, 6, 7, 9}    | {0, 1, 3, 5, 6, 8, 9} | {1, 6}            | {}     |
| $C = 6$ | $R^{\max} = 6$  | {0, 2, 3, 4, 6, 7, 8, 9} | {0, 2, 5, 6, 7, 9}    | {0, 1, 3, 5, 6, 8, 9} | {1}               | {}     |
|         | $R^{\max} = 10$ | {0, 2, 3, 4, 6, 7, 8, 9} | {0, 1, 2, 5, 6, 7, 9} | {0, 1, 3, 5, 6, 8, 9} | {}                | {}     |

**Table 5.1:** Output track lists with different parameters

For each configuration, the table shows what are the detections that the output track lists have accumulated. The two last columns represent the *galleries* that the algorithm assimilate to fictive labels, which are definitely re-identification errors since only three vehicles are present in the scenario. In addition, the cells with green background color are related to the configurations with the *image score* metric consideration.

For  $\{C = 3, R^{\max} = 3\}$  without *image score*, the algorithm struggles to properly reject assignments to the wrong labels, leading to multi-view representation *galleries* composed of different vehicles. Same problem occurs with more cameras ( $C = 6$ ): the final track lists are mixing different labeled



vehicle views when the *image score* quality estimator is not taken into account when assigning to the database. It is due to the fact that images with low resolution are not punished and play the same role as images with good quality during the construction of the *attribution costs*.

The impact of the number of cameras  $C$  compared to the *gallery* size  $R^{\max}$  is similar. In both cases it results in increasing the number of pair comparisons, which leads to better chances of assigning a detection to its correct stored vehicle in the database. With the optimal conditions ( $\{C = 6, R^{\max} = 10\}$  and using *image score*), the algorithm is able to perfectly build the track list of each vehicle.

To summarize the impact of each parameter on the output track lists:

- The consideration of *image score* (in this case, the image resolution) helps to avoid construction of *galleries* with mixed vehicle representations.
- Increasing the number of cameras  $C$  that are capturing the scenes results in better chances of making correct assignments and reduces the number of incorrect *gallery* creations.
- Increasing the *galleries* size  $R^{\max}$  is roughly equivalent to increasing the number of cameras.

Despite the correct vehicle re-identification results on this rather small dataset, the algorithm might encounter two major issues.

First, once a detection has been wrongly assigned to a *gallery*, the *gallery* will store its corresponding images along differently labeled vehicle representations, resulting in somehow a multi-vehicles track list. If in a future scene two vehicles represented by this *gallery* are present at the same time on the parking lot, only one vehicle can be assigned again and the other will end up in another *gallery*, corrupting the database progressively.

Second, if a detection is not correctly assigned to its corresponding *gallery*, it can create a new *gallery* that can slowly result in two parallel representations of the same vehicle that would occupy more space in the database by splitting a track list in two.

To solve these mentioned issues, the algorithm can be improved to include a *gallery* verification process that attempts to split the content of a *gallery* by comparing its vehicle representations in order to cluster similar images. For the other issue, a *gallery* merging process could be designed to again compare images from different *galleries* and decide whether or not the representations have to be gathered in a single *gallery*.

# Conclusion

The objectives of this thesis were to design a multi-cameras vehicle re-identification algorithm for a parking lot scenario and to study the interest of using several cameras to increase the performance.

In order to accomplish these objectives, the state-of-the-art methods for vehicle re-identification were analyzed to list the advantages and drawbacks of each approach (chapter 1). The interest for the deep learning and more precisely the similarity learning (section 1.4) was justified by the fact that this approach allows a robust vehicle images comparison with different camera viewpoints.

Before presenting the siamese network, the parking lot use case was discussed in more detail and a list of specifications was developed to narrow the scope of the scenario possibilities (chapter 2). It has been decided that the re-identification solution would not handle strong illumination variations but would be able to use and combine the viewpoints of multiple cameras while ignoring any license plate information. In addition, a parking lot handcrafted dataset has been recorded and annotated in order to quantify the algorithm performance on a real parking lot scenario with 3 different vehicles, 6 cameras, 10 scenes for a total of 132 individual vehicle crops.

A siamese network has been assembled and trained in chapter 3 to be the image comparison tool of the reID algorithm. The architecture and its loss function were designed to provide a similarity score given a pair of images as input. With an optimal binary decision process discussed in section 5.1, it is able to classify correctly 76.06% of the test pairs.

However this first version of the network was performing poorly on the parking lot dataset with a classification score of 53.5% due to the strong visual similarities of the 3 vehicles present in the scenario (dataset limitations detailed in section 2.2). To overcome this issue, the network was trained again on a mix of both datasets to avoid overfitting on the first while increasing the results for parking lot image pairs. Since the handcrafted dataset is used both for training and testing, this second training phase was interrupted earlier to obtain a classification score of 72.22% on the parking lot dataset and therefore imitate the performance of a non-overfitting model.

The multi-cameras reID algorithm proposed in chapter 4 is described at each level:

- A database structure was created to store vehicle representations and track them through the evolving scene.
- A metric named *image score* was presented and offers a way to quantify the quality of an image according to several specifications (section 4.3). This quantity is indicating to the algorithm which information seems to be the more pertinent or which information should not be trusted because it does not meet the quality criteria.
- Two quantities were defined in order to compare vehicles that have been captured by multiple cameras such as the *pair score* and *matching matrix*. It allows to summarize a comparison of  $N \times M$  RGB images into  $N \times M$  scalar values.

- The assignment to the database process was expressed by defining the *attribution cost* which is using on one hand the scalar values of the *matching matrix* and the *image score* metric on the other hand to make a decision according to the quality of the presented images.
- Once all the *attribution costs* are computed, the assignment choice is solved using the Hungarian method. If a vehicle is matched with a previous detection, the vehicle is considered to be re-identified. Otherwise, the vehicle is considered to be never seen before and its representations are stored under a new label.

The full algorithm was applied to the parking lot dataset to observe its performance and discuss the impact of the number of cameras, the size of the database and the importance of using a quality metric (based on image resolution for the experiment) to select the information.

At first, the cumulative mean *attribution cost* was considered since it represents the quality of vehicle assignments through the whole scenario. The results indicate that increasing the number of cameras and the number of representations per vehicle stored in the database is greatly contributing to minimizing the cumulative mean *attribution cost*. However it also highlights a limitation: if one tries to place more and more cameras to capture a maximum of viewpoints, the cumulative quantity will not decrease accordingly. Indeed, the visual aspect of a vehicle is not an unlimited source of information when captured by cameras which are observing the parking lot.

In a second time, the output track lists of each vehicle in the database were studied. The consideration of the *image score* to assign the new detections was shown to avoid the construction of vehicle representations which are wrongly mixing differently labeled vehicles. On the other hand, the increase of used cameras to observe the parking lot or the increase of the database size helps to reduce the number of cases where a new vehicle label is wrongly created and assigned to the database.

Finally, several ways of improvement are identified:

- One could propose a solution to handle the mixed label vehicle representations issue and the parallel competition of two multi-view representations of the same vehicle by conceiving a splitting/merging process as discussed in section 5.2.
- One could construct an annotated parking lot dataset with more vehicle diversity, occlusions, illumination variation, background diversity and so on. It would allow better research in the domain by breaking the current dataset limitations.
- Since the selection of information based on a quality metric has been shown to improve the assignment process and since the visual aspect of a vehicle on a parking lot seems to be a limited source of information, one could consider other sources of information in addition to RGB images with specific sensors in the environment. A dedicated model could be trained to select and weight all the sources of information according to the current situation and take a decision to assign the abstract vehicle representation to the database.

# Bibliography

- [1] Inc. Gartner. *2 Trends on the Gartner Hype Cycle for Artificial Intelligence, 2020*. <https://www.gartner.com/smarterwithgartner/2-megatrends-dominate-the-gartner-hype-cycle-for-artificial-intelligence-2020/>. (Accessed on 06/16/2021).
- [2] Yihang Lou et al. “VERI-Wild: A Large Dataset and a New Method for Vehicle Re-Identification in the Wild”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.
- [3] [OC] Car color distribution broken down by production year (Poland) : dataisbeautiful. [https://www.reddit.com/r/dataisbeautiful/comments/j7yfqb/oc\\_car\\_color\\_distribution\\_broken\\_down\\_by/](https://www.reddit.com/r/dataisbeautiful/comments/j7yfqb/oc_car_color_distribution_broken_down_by/). (Accessed on 05/11/2021).
- [4] Hongbo Wang, Jiaying Hou, and Na Chen. “A Survey of Vehicle Re-Identification Based on Deep Learning”. In: *IEEE Access* 7 (2019), pp. 172443–172469. DOI: [10.1109/ACCESS.2019.2956172](https://doi.org/10.1109/ACCESS.2019.2956172).
- [5] Xinchun Liu et al. “A Deep Learning-Based Approach to Progressive Vehicle Re-identification for Urban Surveillance”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 869–884. ISBN: 978-3-319-46475-6.
- [6] Hongye Liu et al. “Deep Relative Distance Learning: Tell the Difference Between Similar Vehicles”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2167–2175.
- [7] Yihang Lou et al. “VERI-Wild: A Large Dataset and a New Method for Vehicle Re-Identification in the Wild”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3235–3243.
- [8] Milind Naphade et al. “The 2019 AI City Challenge”. In: *Proc. CVPR Workshops*. 2019, pp. 452–460.
- [9] T. Ojala, M. Pietikainen, and T. Maenpää. “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (2002), pp. 971–987. DOI: [10.1109/TPAMI.2002.1017623](https://doi.org/10.1109/TPAMI.2002.1017623).
- [10] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: [10.1109/CVPR.2005.177](https://doi.org/10.1109/CVPR.2005.177).
- [11] D.G. Lowe. “Object recognition from local scale-invariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2. DOI: [10.1109/ICCV.1999.790410](https://doi.org/10.1109/ICCV.1999.790410).
- [12] Athanasios Voulodimos et al. “Deep Learning for Computer Vision: A Brief Review”. In: *Computational Intelligence and Neuroscience* 2018 (Feb. 2018), pp. 1–13. DOI: [10.1155/2018/7068349](https://doi.org/10.1155/2018/7068349).

- 
- [13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: [10.1145/3065386](https://doi.org/10.1145/3065386).
- [14] *Rectifier (neural networks)* - Wikipedia. [https://en.wikipedia.org/wiki/Rectifier\\_\(neural\\_networks\)](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)). (Accessed on 05/13/2021).
- [15] *Vanishing gradient problem* - Wikipedia. [https://en.wikipedia.org/wiki/Vanishing\\_gradient\\_problem](https://en.wikipedia.org/wiki/Vanishing_gradient_problem). (Accessed on 05/13/2021).
- [16] Ross Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: [10.1109/CVPR.2014.81](https://doi.org/10.1109/CVPR.2014.81).
- [17] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: [1409.1556](https://arxiv.org/abs/1409.1556) [cs.CV].
- [18] Christian Szegedy et al. “Going deeper with convolutions”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9. DOI: [10.1109/CVPR.2015.7298594](https://doi.org/10.1109/CVPR.2015.7298594).
- [19] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: [10.1109/CVPR.2016.91](https://doi.org/10.1109/CVPR.2016.91).
- [20] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: 7 (Dec. 2015).
- [21] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: (Feb. 2015).
- [22] Zhongdao Wang et al. “Orientation Invariant Feature Embedding and Spatial Temporal Regularization for Vehicle Re-identification”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 379–387. DOI: [10.1109/ICCV.2017.49](https://doi.org/10.1109/ICCV.2017.49).
- [23] Alejandro Newell, Kaiyu Yang, and Jia Deng. “Stacked Hourglass Networks for Human Pose Estimation”. In: vol. 9912. Oct. 2016, pp. 483–499. ISBN: 978-3-319-46483-1. DOI: [10.1007/978-3-319-46484-8\\_29](https://doi.org/10.1007/978-3-319-46484-8_29).
- [24] Yanzhu Zhao et al. “Structural Analysis of Attributes for Vehicle Re-Identification and Retrieval”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.2 (2020), pp. 723–734. DOI: [10.1109/TITS.2019.2896273](https://doi.org/10.1109/TITS.2019.2896273).
- [25] Jinjia Peng et al. “Learning multi-region features for vehicle re-identification with context-based ranking method”. In: *Neurocomputing* 359 (2019), pp. 427–437. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2019.06.013>. URL: <https://www.sciencedirect.com/science/article/pii/S0925231219308409>.
- [26] Pedro Antonio Marín-Reyes et al. “Unsupervised Vehicle Re-identification Using Triplet Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 166–1665. DOI: [10.1109/CVPRW.2018.00030](https://doi.org/10.1109/CVPRW.2018.00030).
- [27] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: [1406.2661](https://arxiv.org/abs/1406.2661) [stat.ML].
- [28] Kai Lv et al. “Pose-Based View Synthesis for Vehicles: A Perspective Aware Method”. In: *IEEE Transactions on Image Processing* 29 (2020), pp. 5163–5174. DOI: [10.1109/TIP.2020.2980130](https://doi.org/10.1109/TIP.2020.2980130).
- [29] Zhedong Zheng et al. “Going Beyond Real Data: A Robust Visual Representation for Vehicle Re-identification”. In: *2020 IEEE/Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 2550–2558. DOI: [10.1109/CVPRW50498.2020.00307](https://doi.org/10.1109/CVPRW50498.2020.00307).
- [30] Xiu-Shen Wei et al. “Coarse-to-Fine: A RNN-Based Hierarchical Attention Model for Vehicle Re-identification”. In: June 2019, pp. 575–591. ISBN: 978-3-030-20889-9. DOI: [10.1007/978-3-030-20890-5\\_37](https://doi.org/10.1007/978-3-030-20890-5_37).
-

- [31] Xinyu Zhang et al. “Part-Guided Attention Learning for Vehicle Instance Retrieval”. In: *IEEE Transactions on Intelligent Transportation Systems* (2020), pp. 1–13. DOI: [10.1109/TITS.2020.3030301](https://doi.org/10.1109/TITS.2020.3030301).
- [32] Ramprasaath R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626. DOI: [10.1109/ICCV.2017.74](https://doi.org/10.1109/ICCV.2017.74).
- [33] R. Hadsell, S. Chopra, and Y. LeCun. “Dimensionality Reduction by Learning an Invariant Mapping”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 2. 2006, pp. 1735–1742. DOI: [10.1109/CVPR.2006.100](https://doi.org/10.1109/CVPR.2006.100).
- [34] Weifeng Ge et al. “Deep Metric Learning with Hierarchical Triplet Loss: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VI”. In: Sept. 2018, pp. 272–288. ISBN: 978-3-030-01230-4. DOI: [10.1007/978-3-030-01231-1\\_17](https://doi.org/10.1007/978-3-030-01231-1_17).
- [35] Xinchun Liu et al. “PROVID: Progressive and Multimodal Vehicle reIdentification for Large-Scale Urban Surveillance”. In: *IEEE Transactions on Multimedia* PP (Sept. 2017). DOI: [10.1109/TMM.2017.2751966](https://doi.org/10.1109/TMM.2017.2751966).
- [36] Jianqing Zhu et al. “Joint Feature and Similarity Deep Learning for Vehicle Re-identification”. In: *IEEE Access* 6 (2018), pp. 43724–43731. DOI: [10.1109/ACCESS.2018.2862382](https://doi.org/10.1109/ACCESS.2018.2862382).
- [37] *Softmax function* - Wikipedia. [https://en.wikipedia.org/wiki/Softmax\\_function](https://en.wikipedia.org/wiki/Softmax_function). (Accessed on 05/18/2021).
- [38] Yan Bai et al. “Group-Sensitive Triplet Embedding for Vehicle reIdentification”. In: *IEEE Transactions on Multimedia* 20.9 (2018), pp. 2385–2399. DOI: [10.1109/TMM.2018.2796240](https://doi.org/10.1109/TMM.2018.2796240).
- [39] Yiheng Zhang, Dong Liu, and Zheng-Jun Zha. “Improving triplet-wise training of convolutional neural network for vehicle re-identification”. In: *2017 IEEE International Conference on Multimedia and Expo (ICME)*. 2017, pp. 1386–1391. DOI: [10.1109/ICME.2017.8019491](https://doi.org/10.1109/ICME.2017.8019491).
- [40] Epic Games. *Unreal Engine*. Version 4.22.1. Apr. 25, 2019. URL: <https://www.unrealengine.com>.
- [41] François-Xavier INGLESE - Phd Researcher Assistant - Université catholique de Louvain / LinkedIn. <https://be.linkedin.com/in/françois-xavier-inglese-721a5825>. (Accessed on 08/21/2021).
- [42] Kaiming He et al. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: [10.1109/ICCV.2017.322](https://doi.org/10.1109/ICCV.2017.322).
- [43] A. Neubeck and L. Van Gool. “Efficient Non-Maximum Suppression”. In: *18th International Conference on Pattern Recognition (ICPR’06)*. Vol. 3. 2006, pp. 850–855. DOI: [10.1109/ICPR.2006.479](https://doi.org/10.1109/ICPR.2006.479).
- [44] H. W. Kuhn and Bryn Yaw. “The Hungarian method for the assignment problem”. In: *Naval Res. Logist. Quart* (1955), pp. 83–97.
- [45] Yijun Qian et al. “ELECTRICITY: An Efficient Multi-camera Vehicle Tracking System for Intelligent City”. In: *2020 IEEE/Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 2511–2519. DOI: [10.1109/CVPRW50498.2020.00302](https://doi.org/10.1109/CVPRW50498.2020.00302).
- [46] Meng Zheng et al. *Re-Identification with Consistent Attentive Siamese Networks*. 2019. arXiv: [1811.07487 \[cs.CV\]](https://arxiv.org/abs/1811.07487).
- [47] Tsung-Wei Huang et al. “Multi-View Vehicle Re-Identification using Temporal Attention Model and Metadata Re-ranking”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2019.
- [48] *VCIP 2019, Grand Challenges on Vehicle reIdentification*. [http://www.multimediauats.org/VCIP\\_2019\\_V1/VreID\\_challenges.html](http://www.multimediauats.org/VCIP_2019_V1/VreID_challenges.html). (Accessed on 05/11/2021).



- [49] 10 *Visual embeddings - Deep Learning for Vision Systems*. <https://livebook.manning.com/book/grokking-deep-learning-for-computer-vision/chapter-10/v-8/219>. (Accessed on 05/11/2021).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN  
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | [www.uclouvain.be/epl](http://www.uclouvain.be/epl)