Titre: Title:	A Design Approach for Efficient Inter-Camera Vehicle Tracking
Auteur: Author:	Mahsa Nazemi Gelian
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Nazemi Gelian, M. (2022). A Design Approach for Efficient Inter-Camera Vehicle Tracking [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10449/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

A Design Approach for Efficient Inter-Camera Vehicle Tracking

MAHSA NAZEMI GELIAN
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

A Design Approach for Efficient Inter-Camera Vehicle Tracking

présenté par **Mahsa NAZEMI GELIAN**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Tarek OULD-BACHIR, président
Gabriela NICOLESCU, membre et directrice de recherche
Guillaume-Alexandre BILODEAU, membre et codirecteur de recherche
Heng LI, membre

DEDICATION

To my beloved parents,
for their everlasting unconditional love and support.

ACKNOWLEDGEMENTS

I would like to thank my research supervisors, Prof. Gabriela Nicolescu, and Prof. Guillaume-Alexandre Bilodeau, for giving me the opportunity to do my research under their supervision. This project would have not been possible without their continuous support and precious guidance.

I would like to extend my sincere thanks to the entire team at Cysca Technologies, for providing me the opportunity to do research and gain priceless experiences in industrial contexts. I am also extremely thankful to Mitacs for providing the funding for this research.

I would like to express my deepest appreciation to Prof. Vahid Partovi Nia for his unwavering support, and invaluable advice. His constructive criticism and comments on this thesis are priceless.

Thanks should also go to Dr. Felipe Gohring de Magalhaes, as the second reader of this thesis. I am grateful for his valuable comments on this thesis.

My sincere thanks go to the members of the jury for taking time to review my thesis and to give valuable feedback.

I also want to express my gratitude to my colleagues at the HES lab for being supportive. Finally, I want to express my heartfelt thanks to my dear friend, Hamed, and my dearest family for their endless love, support, and encouragement.

RÉSUMÉ

Avec l'avènement de l'apprentissage en profondeur et de la vision par ordinateur, de nombreuses recherches ont été menées sur le développement de systèmes de transport intelligents efficaces. De plus, le suivi et la réidentification des véhicules à travers un réseau de caméras sont parmi les problèmes les plus attrayants de la conception de systèmes de gestion du trafic. Les algorithmes d'apprentissage profond assisté par modèle sont normalement utilisés pour s'attaquer à ces tâches. D'autre part, atteindre l'efficacité de ces algorithmes est la limite de leur utilisation dans les applications du monde réel. Ce travail se concentre sur la présentation d'un traqueur inter-caméras (TIC) efficace en temps réel de véhicules qui est applicable en milieu urbain avec diverses structures de route et de chemin. La méthode que nous proposons utilise des caractéristiques d'apparence ainsi que des informations spatio-temporelles pour associer des trajectoires à travers plusieurs caméras. Pour ce faire, un modèle basé sur des réseaux de neurones convolutionnels est utilisé pour créer les descripteurs basés sur l'apparence des trajectoires. De plus, pour faire correspondre les trajectoires des candidats, des modèles de liaison de caméra basés sur la trajectoire sont définis et utilisés pour leur appliquer les contraintes spatio-temporelles. Ainsi, en appliquant ces contraintes, le nombre de paires de trajectoires appariables potentielles est considérablement réduit puis la matrice de distance est calculée pour les paires restantes.

Ces distances sont calculées sur la base des descripteurs d'apparence de trajectoire et les trajectoires avec la plus petite distance sont appariées à l'aide d'un algorithme de recherche gourmand. Par conséquent, la trajectoire complète de chaque véhicule à travers le réseau de caméras est construite et attribuée à un identifiant global unique. Nous utilisons un ensemble de deux modèles basés sur ResNet pré-formés qui sont améliorés pour extraire les caractéristiques de trajectoire. En conséquence, les trajectoires sont décrites par des caractéristiques d'apparence plus discriminantes, ce qui conduit à une amélioration de l'efficacité des TIC. Nous proposons également d'améliorer l'association de trajectoires en incorporant des voisins de trajectoires dans la mesure de distance au lieu de se fier uniquement à la distance euclidienne des trajectoires. Les méthodes TIC existantes ont une grande complexité de calcul et exigent des ordinateurs puissants. Les méthodes normalement proposées pour rendre les TIC efficaces en termes de temps et d'utilisation de la mémoire ont un impact négatif sur leur efficacité. Nous introduisons une méthodologie pour améliorer les performances des TIC existantes en réduisant les frais généraux d'exploitation qui leur permettent de s'exécuter sur des appareils à ressources limitées. En appliquant cette méthodologie à nos TIC, nous pouvons constater une amélioration significative des performances en termes de réduction du

temps de traitement et de l'utilisation de la mémoire.

Dans le monde réel, les conditions météorologiques défavorables (par exemple neige, pluie, vent, etc.) ont un impact significatif sur les performances des TIC. Ainsi, il est important de valider notre TIC dans de telles conditions météorologiques pour avoir une attente réaliste de l'TIC dans les scénarios du monde réel. Motivés par cet objectif, nous avons créé un ensemble de données synthétiques simulant des conditions météorologiques difficiles afin de pouvoir valider et comparer diverses TIC dans de telles conditions météorologiques. À notre connaissance, c'est la première fois qu'un tel ensemble de données est présenté, basé sur les données AIC20, contenant les effets synthétiques de la neige et de la pluie. Notre TIC proposée présente une efficacité améliorée sur le benchmark AIC20, en particulier en termes de score IDF1. De plus, le TIC que nous proposons obtient une efficacité d'utilisation des ressources élevée en réduisant la consommation de mémoire de trois ordres de grandeur et en augmentant la vitesse de traitement de 1.8×, atteignant 22 images par seconde de vitesse de traitement. Pour les travaux futurs, nous pouvons évaluer l'impact de l'utilisation de modèles Re-ID compressés sur le débit du système et l'amélioration de l'utilisation de la mémoire.

ABSTRACT

With the advent of deep learning and computer vision, there has been tremendous research on developing efficient intelligent transportation systems. Moreover, Vehicle tracking and re-identification across a network of cameras are among the most appealing problems of designing traffic management systems. Model-aided deep learning algorithms are normally used to tackle these tasks. On the other hand, the efficiency of these algorithms is the bottleneck for using them in the real-world application. This work focuses on presenting an efficient real-time Inter-Camera Tracker (ICT) of vehicles that is applicable in urban settings with various road and path structures.

Our proposed method uses appearance features as well as spatial-temporal information to associate trajectories across multiple cameras. To accomplish that, a model based on convolutional neural networks is utilized to create the appearance-based descriptors of the trajectories. Furthermore, for matching the candidates' trajectories, trajectory-based camera link models are defined and used to apply the spatial-temporal constraints on them. Thus, by applying these constraints, the number of potential matchable trajectory pairs are reduced substantially and then the distance matrix is calculated for the remaining pairs. These distances are calculated based on the trajectory appearance descriptors and the trajectories with the smallest distance are matched using a greedy search algorithm. Therefore, the complete trajectory of each vehicle across the cameras' network is constructed and assigned to a unique global ID. We use an ensemble of two pre-trained ResNet-based models that are improved to extract the trajectory features. As a result, the trajectories are described by more discriminating appearance features which leads to improvement of ICT effectiveness. We also propose to improve trajectory association by incorporating neighbors of trajectories in distance measurement instead of relying solely on the Euclidean distance of trajectories.

The existing ICT methods have high computational complexity, and demand powerful computers. The methods that are normally proposed to make the ICT efficient in terms of time and memory usage, have a negative impact on their effectiveness. We introduce a methodology to enhance the existing ICTs' performance by reducing operation overhead that enables them to execute on resource-constrained devices. By applying this methodology to our ICT, we can see a significant performance improvement in terms of reducing processing time and memory usage.

In the real-world, adverse weather conditions (e.g. snowy, rainy, windy, etc.) have a significant impact on the ICTs' performance. Thus, it is important to validate our ICT in such

weather conditions to have a realistic expectation of the ICT in the real-world scenarios. Motivated by this goal, we created a synthetic dataset simulating difficult weather conditions to be able to validate and compare various ICTs in such weather conditions. To the best of our knowledge, this is the first time such dataset is presented which is based on the AIC20 data, containing synthetic snow and rain effects.

Our proposed ICT exhibits improved effectiveness on the AIC20 benchmark, particularly in terms of IDF1 score. Moreover, our proposed ICT obtains high resource usage efficiency by reducing memory consumption by a three orders of magnitude, and increasing processing speed by 1.8×, achieving 22 frames per second of processing speed. For future works, we can evaluate the impact of using compressed Re-ID models on the system throughput and memory usage improvement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

ITS	Intelligent Transportation System
MCT	Multi-Camera Tracking
ICT	Inter-Camera Tracking
SCT	Single-Camera Tracking
Re-ID	Re-IDentification
CNN	Convolutional Neural Networks
IDF1	IDentification F1
IDP	IDentification Precision
IDR	IDentification Recall
IDTP	IDentity match True Positives
IDFN	IDentity match False Negative
IDFP	IDentity match False Positive
FPS	Frame Per Second

CHAPTER 1 INTRODUCTION

Multi-camera vehicle tracking systems are of great interest in the modern intelligent world because of their advantages in building smart cities. In this chapter we give our motivations of our project and present our research objectives and contributions.

1.1 Context and Motivation

Computer vision is a field of computer science that focuses on replicating parts of the human vision system enabling computers to comprehend and interpret their environment visually [1]. This field profits from artificial intelligence, and in conjunction with deep learning models surpasses human visual abilities in many areas such as face recognition [2] and processing live actions in a crowded scene. This is particularly useful for interpreting crowded areas, whereas human eyes are not capable of processing huge amount of information [3]. The applications of computer vision provide substantial improvements in many industries (e.g. transportation, healthcare, manufacturing, agriculture, and retail), making it an indispensable part of technological development and digital transformation [4].

In recent years, with the ever-increasing tendency towards creating smart cities, the demands of Intelligent Transportation System (ITS) have quickly risen. ITS is a critical field for enhancing the efficiency, effectiveness, and safety of transportation [4]. Some of the most popular computer vision applications in ITS are self-driving cars, vehicle and pedestrian detection, parking occupancy detection, and traffic flow analysis. The demands of Multi-Camera Tracking (MCT) have been rapidly increased in recent years, thank to the growth of traffic camera networks [5], helping to design better traffic management systems and improve road safety [4].

MCT is a complicated task that requires various computer vision algorithms and techniques. Many methodologies have addressed this task, and designed effective tracking systems, using deep learning models-aided algorithms [6], [7], [5], [8]. However, these methods have high computational complexity, and demand powerful computers. One of the high resource demanding part of MCTs is Inter-Camera Tracking (ICT), which is responsible for re-identifying vehicles in different cameras, to match the trajectories belonging to the same vehicle, and construct complete trajectories of the vehicles. To the best of our knowledge, efficient methods for reducing the overhead of these methods are not easily available in literature. We present an ICT method inspired by a state-of-the-art work, enhancing its effectiveness using

a modified re-identification model and trajectory matching sub-task. Then, we introduce a methodology to reduce the operation overhead of the ICT, in terms of the execution time and memory consumption, without having a negative impact on their effectiveness.

1.2 Problem Statement

Intelligent transportation systems is currently an active research area thanks to the rapid advancements in computer vision and deep learning algorithms [9]. Vehicle tracking and re-identification through a network of cameras is one of the appealing problems in this area, because of their contribution in providing safety and security. The approaches toward addressing these tasks are divided into two categories [10]: (i): The vehicle ID (e.g., license plate number) along with location and time information are continuously sent to a control center and the tracking is performed using the Global Positioning System (GPS), (ii): The vehicle is identified when it enters the field-of-view of the designated device. The devices are surveillance cameras, radio-frequency identification, and inductive loops.

Among all these technologies, surveillance cameras are more desirable, because the cameras that are already installed in parking lots and roadsides are used without additional charges. Even if there are not already cameras installed in the place, there is no need for complex construction work for their installation, as opposed to using other devices that require more costs and installation efforts [10].

The goal of this project is to bring a solution enabling to use surveillance camera videos to establish completed trajectories of the observed vehicles in an area that is monitored by a network of cameras. Thus, the main tasks accomplished in this project are to determine whether the given vehicle observed in one camera, has appeared in the other cameras or not, and to match the trajectories of the same vehicles across the camera network. Some of the difficulties that are observed to perform this work are as follows:

- **Intra-class variability:** the same vehicle looks visually different over the camera network, because of the variety of scales and shapes from different view angles. Furthermore, the lighting effects may change the color and illumination of the vehicle in different cameras [11], [12]. Figure Figure 1.1 shows an example of intra-class variability caused by viewpoint variations, where we can see appearance variation in the three images that are recorded by different cameras from the front side, back side, and rear side of the same bus.
- **Inter-class similarity:** different vehicles look visually similar, especially when they are observed from the same view angle. The problem is due to the similarity in shape

Figure Figure 1.1 An example of intra-class variability caused by viewpoint variations [10]. From left to right, the images show front side, back side, and rear side of the same bus, respectively. We can see the appearance variation in these images that are captured by three different cameras.

and appearance between some vehicles that are produced by different manufacturers. For example, they may have a similar rear or front side [10]. An example of this is illustrated in Figure Figure 1.2. In this figure the images from the back side of three different cars are shown, that share similar appearance features.

- **Partial occlusion:** the output of the appearance feature descriptor of a vehicle is corrupted when some part of the vehicle is covered by another vehicle or any other objects. In Figure Figure 1.3, we can see an example of partial occlusion, where the white car is partially occluded by another vehicle. This problem often appears, when the scene is crowded [13], [14].

- **Resolution variation:** some of the cameras that are already installed on roadsides and parking lots are old, so they capture low-resolution frames. Using such frames to distinguish vehicles is difficult, because they lack of discriminatory details [10]. The same problem appears when there are difficult weather conditions such as snow and rain. Figure Figure 1.4 illustrates a frame recorded in a snowy weather condition, in

Figure Figure 1.2 An example of inter-class similarity [10]. Images captured from the back side of three different cars, that share similar appearance features.

Figure Figure 1.3 Example of partial occlusion. The white car is partially occluded by the front car.

Figure Figure 1.4 Example of vehicle images recorded in snowy weather, in which extracting appearance features including discriminatory details is difficult.

which the appearance of the vehicles are effected by the snow flakes.

Matching trajectories based on the appearance-based descriptors of vehicles cannot solely provide an effective tracking system, because many vehicles share the same models and different vehicle with the same models can look highly similar [15]. Therefore, spatial-temporal information and constraints need to be considered as well. However, the cross-camera spatial-temporal constraints are dependent on the camera network configurations. Thus, a strategy that easily fit to the new environments with different camera configurations is an interest of the industry.

Therefore, in this project we focus on developing a method to establish complete trajectories of the vehicles across multiple cameras by matching the trajectories that belong to the same vehicle and are observed in each camera. Our method performs in real-time speed. Through this project, we address the challenges of providing discriminating appearance features of vehicles, intra-class variability, and inter-class similarity. Furthermore, we focus on resource usage improvement.

There are many methods to approach this problem that we review in Chapter 2. Inspired by [5], we have further extended its methodology to tackle this problem, accomplishing the following tasks:

- Extracting appearance features of the vehicles that were observed on the video frames, i.e., frame-level features, and providing feature vectors that represent trajectories of these vehicles in each single camera, i.e., clip-level features. The appearance features encode the object shape, size, color, etc.

- Applying spatial-temporal constraints related to the environment, camera network configurations and the trajectories information. The goal is to narrow down the number of candidate trajectories that will be compared for association across cameras.
- Matching trajectories across cameras based on their appearance features and spatial-temporal constraints. Then, assigning unique global identifiers to each completed trajectory that is constructed.

1.3 Scope of the Project

This project has been conducted in collaboration with an industrial partner, Cysca Technologies [1], that specializes in providing efficient technological solutions to develop electronic systems and their related embedded software.

Our work is introduced as part of a comprehensive multi-camera vehicle tracking system, which tracks the movement of targets and connects the trajectories across multiple camera views. Researchers consider multi-camera tracking task as either a global detection association [16], [17], or a task consisting of two sub-tasks: (i) Single-Camera Tracking (SCT), which provides the trajectories of the targets in each single camera, and (ii) Inter-Camera Tracking (ICT), to associate the trajectories across multiple cameras [18], [19], [20]. Thus, the latter approach supports modular implementation, so that SCT and ICT are considered as two modules of the MCT. Figure Figure 1.5, shows these two types of approaches for designing a multi-camera tracking. Both approaches are capable of providing MCTs with performance comparable to the state-of-the-art. However, the main advantage of the latter approach over the former one is modularity. Thus, using the second approach, one can alternate any other top-performing methods of SCT and ICT to enhance MCT effectiveness.

We take the advantage of modularity of the second approach to obtain the ultimate MCT. Our work is focused on providing an ICT, and the research on SCT has been conducted as another project.

More particularly, our project is developing a system that is able to process videos from multiple surveillance cameras to provide the complete trajectories across cameras network, given the tracking results in each single camera. Therefore, the input of the system are videos from cameras that are installed above ground level, along with the trajectories of the vehicles that are tracked independently in each camera.We develop a reliable system in terms of both system effectiveness metrics and resource usage.

[1] https://www.cysca.com/en/

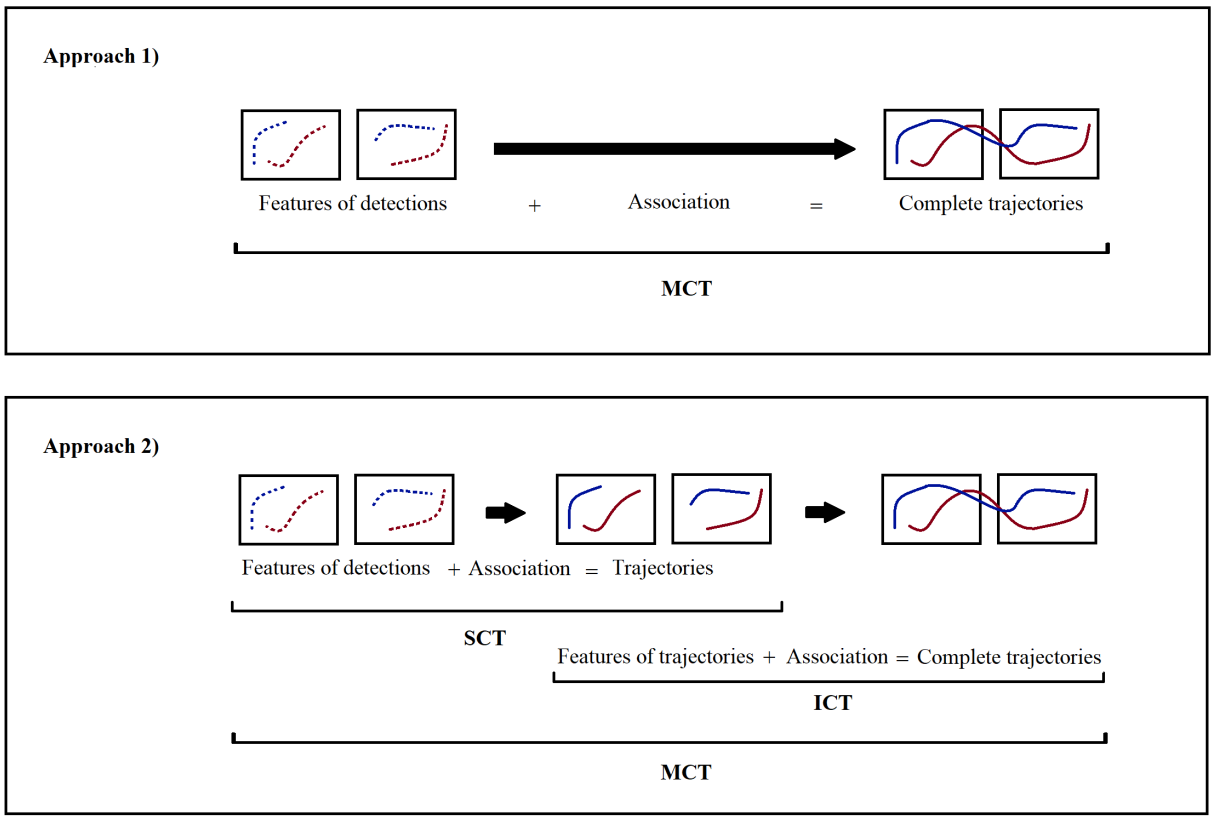


Figure Figure 1.5 Two types of approaches for MCT. In the first approach (upper diagram), there is only one association step; this way, the complete trajectories are constructed based on a global association of the detected vehicles across all cameras. In the second approach (lower diagram), the MCT task consists of two modules of SCT and ICT. The first module, i.e., SCT, is performed for each camera independently. In this module, the vehicles are detected and localized throughout the frames and the trajectory of each vehicle is created, based on the detections association. Then, the outputs of this module are fed to the second module i.e., ICT, to perform trajectory association, and create the complete trajectories.

The system work in urban scenes, such as roads and parking lots, in which various vehicle models and categories (e.g. cars, trucks, buses and vans) travel. The cameras can have overlapping or non-overlapping field-of-views. Our system has the following constraints:

- **Real-time processing:** processing the videos from multiple cameras must be performed in real-time (i.e., more than one frame per second). Here, video processing includes extracting the features of the single camera trajectories, exploiting spatial-temporal information, and trajectories association.
- **Applicability in various environments:** our system must ensure its effectiveness in both outdoor parking lots and roads scenes that have various structures and path constraints.

1.4 Research Objectives and Contributions

Our goal is developing an efficient pipeline for inter-camera vehicle tracking. In this processing pipeline videos from multiple surveillance cameras along with the vehicle trajectories in each camera view are received. Then, these input are processed to provide complete trajectories with real-time processing speed. The proposed ICT must be applicable in urban settings with various road and path structures. Our research objectives can be summarized as follows:

- Selecting an appropriate base pipeline with respect to our system constraints which has a comparable performance to the-state-of-the-art.
- Enhancing the effectiveness of selected pipeline by addressing challenges of inter-class similarity, and intra-class variability while preserving the system throughput.
- Improving the resource usage to speed up the system throughput and reduce the memory footprint.
- Validating our proposed method on an annotated dataset that is commonly used for MCT systems evaluations, and on videos acquired from designated parking lots.
- Validating our proposed method under challenging weather conditions.

Thus, we have made the following contributions to achieve the aforementioned goals:

- The effectiveness of our system is improved in terms of MCT evaluation metrics by (i) revising pairwise distance definition, using k-reciprocal nearest neighbors, in trajectory association, and (ii) using an ensemble model for Re-ID features extraction.

- The system throughput and memory footprint is significantly improved using our proposed resource usage reduction strategy.
- A synthetic annotated dataset with snowy and rainy weather is created to validate our method under difficult weather conditions.

1.5 Document outline

The rest of this document is organized as follows. We introduce basic concepts, and review previous work on the main elements of our project in Chapter 2. In Chapter 3, we present the methods that are used to conduct our project. The implementation details and experimental results are discussed in Chapter 4. We present the conclusion of our work and its limitations in Chapter 5.

CHAPTER 2 DEFINITIONS AND LITERATURE REVIEW

This chapter presents concepts as well as state-of-the-art approaches of the major tasks of our work. The value of addressing time and memory efficiency of the ICTs are also highlighted.

2.1 Definitions

In this section we define the key concepts, that are discussed throughout our thesis to have a better understanding of a MCT.

2.1.1 Single-Camera Tracking (SCT)

A single-camera vehicle tracking system provides the trajectories describing the movement of vehicles in the video frames recorded in one single camera. A SCT assigns a distinct identifier to each detected vehicle at each frame, and maintains these identifiers in the subsequent video frames. Thus, a trajectory is defined as a sequence of detections that belong to the same vehicle in a video, carrying a unique consistent identifier. Figure Figure 2.1, illustrate sample of SCT results in two frames, where two vehicles are located in the frames by a bounding box and assigned to consistent identifiers.

The main task is defining characteristics of the new detections at each frame, and comparing them to the trajectories that have been established so far during the previous frames. To this end, the features of detections and trajectories are extracted, encoding information such as the vehicle appearance, their location, the time of their movement, etc. Then, the association cost for matching every pair of detection and trajectory is calculated and the pairs that resulted in lower cost are matched, and a trajectory is created. An example of part of an association cost is the distance between their appearance feature vectors.

2.1.2 Inter-Camera Tracking(ICT)

As previously mentioned SCTs track the vehicles from one viewpoint of a camera. On the other hand, multi-camera tracking means to track the vehicles that are observed across a network of cameras, i.e., more than one camera, and maintain a unique global identifier for each vehicle. The trajectories associations in these systems are more challenging than in single-cameras, because the appearance of the same vehicle tends to vary due to the view angle variations and environment inconsistency in different camera views. Furthermore,

Figure Figure 2.1 Sample frames of output of a SCT. The two scenes are captured in about 400 frame span. In both scenes two vehicles are located in the images, rounded by a bounding box, and assigned to consistent IDs (the white car is assigned to ID no.9, and the blue car is assigned to ID no.7).

the camera configurations, their position related to each other, and their overlapping and non-overlapping field-of-views make interpreting the spatial and temporal information more difficult than when we have only one camera.

The task of matching trajectories that are recorded on different cameras, based on their appearance and time and location-related information is performed by inter-camera tracking systems.

2.1.3 Re-identification(Re-ID)

In tracking systems, when a target travels through multiple camera views, the recognized target is identified again when it enters another camera view, no matter whether the cameras have overlapping or non-overlapping field-of-views. This process is called re-identification (Re-ID) [21].

Vehicle Re-ID aims to search, locate and track the target vehicles across camera networks, and is a key task in preserving public security. It also serves as a core element in the large-scale vehicle recognition, intelligent transportation, and surveillance video analytic platforms [21].

In some cases license plate numbers could be used for vehicle Re-ID. However, in most cases Re-ID based on license plate numbers is not feasible, because of the privacy concerns, or the low-resolution of the videos recorded by surveillance cameras that prevent recognizing the license plate numbers accurately [21]. Thus, it is necessary to re-identify vehicles without their license plates, and based on the vehicle appearance characteristics.

Appearance-based vehicle Re-ID in inter-camera tracking systems refers to the problem of matching multiple images of the same vehicle under intense variations in appearance, illumination, pose, and viewpoint in different camera views [22]. This is done to identify the same vehicle, in a large scale vehicle trajectories database, and construct the complete trajectories of the vehicles.

2.1.4 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN), is a feed forward neural network composed of three types of neural network layers [21]:

- (i) convolutional layer: these layers are used to learn the feature representation of the input data, by conducting convolution on the input layers. Each layer is composed of multiple convolution kernels to acquire different feature maps [21]. An example of a convolution operation performed on one pixel of an image, is shown in Figure Figure 2.2.
- (ii) pooling layer: these layers are mostly used to reduce the model size, speed up the computations, and enhance the robustness of extracted features [21]. The most popular pooling layers are averagepooling and maxpooling. An example of maxpooling is shown in Figure Figure 2.3.
- (iii) fully-connected layer: in these layers, each node of the layer is connected to all nodes of the previous layer and are used to integrate the features that are extracted in the previous layer (Figure Figure 2.4). These layers serve as a linear transformation that reduces the dimension of the features. Thus, fully-connected layers are often found in the last few layers of a CNN to perform a weighted sum of the extracted features.

A diagram of the basic structure of convolutional neural networks is shown in Figure Figure 2.5. CNNs inherently learn representations of objects in an image when they are trained for classification tasks. The representations are often obtained in the last fully-connected layers, before the final layer in which classification is performed. Hence, CNN backbones can be used to extract deep feature descriptors of vehicles.

2.1.5 k-reciprocal nearest neighbors

The k-reciprocal nearest neighbor is a stricter rule of neighborhood than the ordinary k-nearest neighbors. In a set of samples, two samples are called k-reciprocal nearest neighbors if they are both ranked top-k nearest neighbors when the other sample is taken as the query [24].

2.2 Literature Review

In this section we explore state-of-the-art approaches for the the main tasks of an ICT system, consisting of extracting appearance features of the vehicles for Re-ID, and applying spatial-

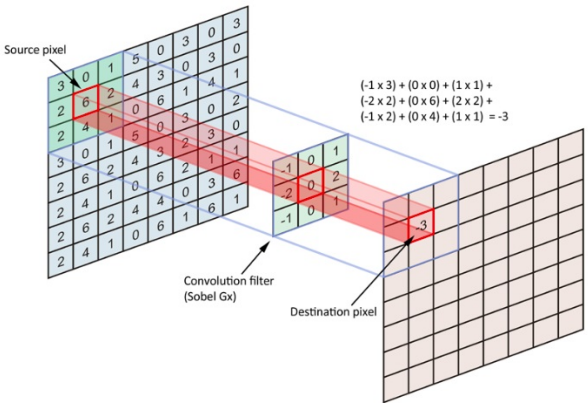


Figure Figure 2.2 The convolution filter slides over the input image and performs its output on the new layer [23].

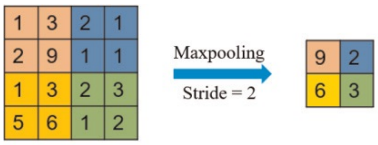


Figure Figure 2.3 An example of maxpooling. The stride parameter, which determines the number of filter movement over the matrix, is set to 2.Thus, the input of 4×4 is divided into four different regions. For the output of 2×2, each element output is the maximum element value in its corresponding color region [21].

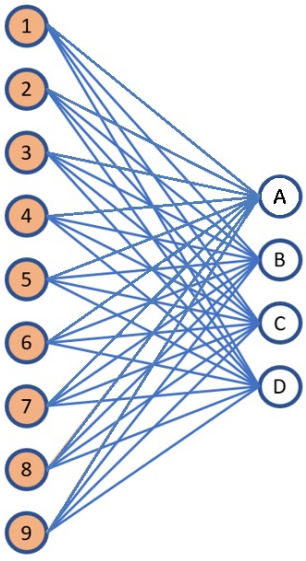


Figure Figure 2.4 A fully-connected layer, in which each node of the layer is connected to all nodes of the previous layer.

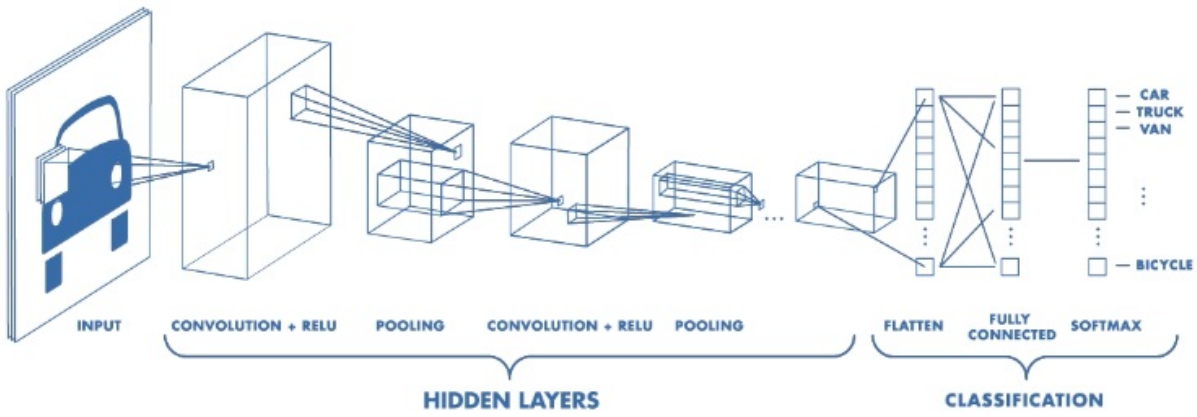


Figure Figure 2.5 The basic structure of Convolutional Neural Networks (CNN) [23]. Each convolution layer is followed by an activation function layer (RELU). T The result of activation function is fed to a pooling layer to reduce the number of nodes. In classification layers, all the 2-dimensional arrays resulting from the last pooling layer are flattened to convert to a long linear vector. This vector is fed to a fully connected layer that represents all the image features. These features are used to classify the image.

temporal constraints through trajectory associations. Then, we discuss the value of efficient implementation of ICTs in terms of time and memory.

2.2.1 Appearance Features for Re-ID

The core of trajectory association is re-identifying the vehicles based on their appearance. The appearance feature of a vehicle image includes color feature, texture feature, shape feature and spatial relationship feature [21].

The appearance feature extraction methods in computer vision are mainly divided into two groups: (i) traditional methods, such as Histogram of Gradient (HOG) [25], [26], scale-invariant feature transform (SIFT) [27], and Local Binary Pattern (LBP) [28]. (ii) deep-learning based methods including different types of neural networks such as CNNs [21]. **HOG:** In this method the image is decomposed into small squared cells. For each cell, a histogram of oriented gradients is computed. Then, the results are normalized using a block-wise pattern, and return a descriptor for each cell [26]. Therefore, this method focuses on edge information. HOG method ignores the effect of color and lighting on the image. In addition, using HOG leads to reduce the characterization dimensions. However, HOG is

sensitive to noise and occlusion [29].

SIFT: In this method first the potential locations for finding features are selected, named keypoints. Then, the orientation of these keypoints are calculated and described as a high dimensional vector, which are the descriptions of the corresponding key-points [27]. Therefore, this method focuses on local information. SIFT method is invariant to rotation, scale and brightness changes, however, it requires large amount of calculation, and it can not provide real-time speed on limited-resources systems [29].

LBP: In this method each pixel of the image is labeled to a binary number i.e., LBP codes, by thresholding the neighborhood of each pixel. Particularly, each pixel is compared with its 8 neighbors, and the neighbors are assigned to 1 if its value is more than the value of the center pixel. Then, the histogram of LBP codes is constructed as the LBP feature [28]. Therefore, this method focuses on texture information. LBP is fast because it consists of simple calculation. It is invariant to light changes, but is sensitive to direction of the image [29].

Deep-learning based methods: With the emergence of deep learning-based vision tasks, vehicle re-identification has been greatly improved. Furthermore, deep learning-based feature descriptors are more robust against lighting variations, scale changes and rotations than conventional feature descriptors. Among the deep-learning based methods, CNN-based visual features representation has recently gained a lot of attention and outperforms other neural networks in the vehicle re-identification field of research [7], [30].

Liu et.al. in [6] studied the impact of using three different CNN backbones on re-identification performance. They conclude that the best performance is achieved when a merge of these backbones are used to represent the vehicle features.

Although, extracting robust and discriminating features for Re-ID has been dominated by CNN-based methods for some time [31], [32], [33], the attention of the community has been recently attracted to transformer-based deep neural networks that achieve state-of-the-art performance on both person and vehicle Re-ID benchmarks as in [34].

Using the fusion of different features for re-identification is also studied, such as fusion of manual features and deep convolutional-based features or fusing the features of different regions of the image [21]. For instance, Li et al. [35] proposed a vehicle Re-ID algorithm based on fusion of extracted features from different regions of the vehicle. In [35], first the attention of each region of the vehicle image is obtained using a region detection algorithm proposed by [36]. Then, the features of each detected region are extracted, and the features are fused to generate new fusion features.

2.2.2 Spatial-temporal Constraints

In traffic monitoring, time and location related information play an important role to enhance the re-identification performance in tracker systems, other than appearance features of the vehicles. A segmented vehicle Re-ID algorithm is proposed by Liu et al. [37], in which spatial and temporal information are utilized to improve the re-identification performance. In [37], the appearance features are first used for the primary screening, then the images that belong to the same vehicle are matched based on their detected license plate numbers. The results are integrated with spatial and temporal information for reordering, which brings in more accurate vehicle matching. Jiang et al. [38] proposed an effective vehicle Re-ID algorithm that performs the primary screening, using the fusion of multiple features including the vehicle image color, the vehicle models, and vehicle appearance features. The results are then reordered and sorted based on spatial-temporal similarity. Wang et al. [39] proposed a vehicle Re-Id architecture which consists of two main components, i.e., the orientation invariant feature embedding and the spatial- temporal regularization. In the first component, local region features are extracted based on the locations of key points, then these features are aligned and combined to form orientation invariant feature representations. A spatial-temporal regularization model is then adopted for refining the retrieval results.

Some scholars [5,6,40–42] consider spatial-temporal constraints by defining camera link models, to reduce the searching and matching space in the ICT [5]. The camera link models are a set of predefined spatial-temporal constraints related to the camera networks and their field-of-views. A reliable camera link model significantly narrows down the candidate set for matching, subsequently, improves the accuracy of cross camera trajectory association [5]. In [41], [42], the camera link models are built based on the transition time between the cameras. This is, for each connected pair of cameras the transition time distribution is determined using the estimated vehicles' speed. Hsu et al. [5] proposed trajectory-based camera link models for multi-camera tracking of vehicles that achieves state-of-the-art MCT performance on AIC19 dataset (IDF1 score of 70.5). They build the camera link models on the basis of the existing driving constraints in the environment, due to the road structures and traffic rules, etc. Particularly, they predefined all the possible trajectories' patterns in each camera, and they propose camera link models by exploiting the spatial-temporal relationships between the trajectories' patterns in different cameras. Since the models are built based on the vehicles' trajectories patterns, it can be used in every environment, such as roads and parking lots, by defining the trajectory patterns that are compatible to that environment. On the contrary, the camera link models proposed in [6] are only practical in the environments that have crossroads, because their proposed strategy is according to the characteristics of

the crossroads. Thus, although, using the proposed strategy for building camera link models in [6] achieves state-of-the-art MCT performance on AIC20 dataset (IDF1 score of 80.9), it cannot be generalized to use in parking lots. Note that the performance results of the methods proposed by [5] and [6] are not comparable because, they have been evaluated on different sets of data with various metadata. There are studies [43], [8] carried out on top of the method presented in [5] that improved its effectiveness, however, none of them addressed the time efficiency and memory footprint of the tracker. Our work is also conducted on top of [5], and we improve time and memory efficiency.

2.2.3 Time and Memory Efficiency

Different solutions have been proposed to design effective ICTs [6, 7, 44]. An important hub for state-of-the-art solutions for this is the annual AI CITY challenges, held by NVIDIA[1]. Although important advances are presented in such conferences, these are mainly concentrated on the effectiveness of the tracker, not on the efficient implementation in terms of execution time and memory consumption. This is due to the nature of targeted solutions: originally these are designed to execute on High Performance Computing (HPC) systems, that don't have resource limitations. Thus, default implementations of ICTs perform well when executing on such environments without resource limitations. Nevertheles, they will face serious challenges to perform as expected when we try to execute them on commonly available systems with limited resources. In addition, future IoT paradigm brings edge computing, in which resources can be scarce, so the same challenges will be emerged in executing current ICTs on edge devices. Thus, measures need to be taken toward reducing reource usage to make existing ICTs executable in this contexts.

One way to increase the efficiency of the exixsiting ICTs is to rely on images with lower resolution to minimize the required resources for image processing algorithms. However, this would affect the system's effectiveness. Another possibility is to analyze the required resources to execute the application and try to adapt the algorithm to it. However, it brings additional computational complexity and potentially reduces algorithm efficiency. A MCT is proposed in [45], that uses edge computing and low-power communication, but it performs only when the fields of views of neighboring cameras overlap partially. To the best of our knowledge, there is no existing ICT system that is memory and time efficient, while using deep learning models and providing state-of-the-art performance.

Further, in the existing studies on designing ICTs, the required equipment (in terms of memory, hard drive, CPU capacity) and the speed of process are not reported quantitatively.

[1] https://www.aicitychallenge.org/

Such report would ease understanding points to improve in the algorithm. These challenges motivate us to address the time and memory efficiency of our ICT, and report the required resources to run our ICT. More importantly, we present a methodology to enhance the existing ICTs efficiency. In this method we identify the bottlenecks of an implemented ICTs and indicate points of improvement in terms of execution time and memory consumption, We validated the proposed methodology by applying it on our ICT, which resulted in drastically reduction in both execution time and memory footprint.

CHAPTER 3 DESIGN APPROACH FOR EFFICIENT INTER-CAMERA VEHICLE TRACKING

This chapter introduces the developed ICT method. In Section 3.1, we introduce the selected base pipeline and discuss its components in detail. Then, in Section 3.2, we highlight the strengths and weaknesses of the selected methods and present our modification to improve the ICT pipeline effectiveness. In Section 3.3 we illustrate our resources usage improvement strategy to enhance the efficiency of our proposed ICT. In the last section a summary of the chapter is presented.

3.1 Inter Camera Tracking Method

The ultimate goal of MCT is to track the movement of the targets in each camera and provide complete trajectories for each target passing across multiple cameras [5]. The cameras may have overlapping or non-overlapping field of views. A MCT includes two main components: i) single-camera tracking (SCT), to track vehicles locally in each single camera, and ii) inter-camera tracking (ICT), to connect the trajectories across multiple cameras. Mores specifically, ICT receives the SCT's results as input to generate complete trajectories across cameras based on visual re-identification (Re-ID) features and spatial-temporal constraints. Therefore, we divide the ICT component into two tasks: i) appearance-based Re-ID features extraction, and ii) exploiting spatial-temporal constraints for cross cameras trajectory matching according to the trajectories' Re-ID features distances [5]. We rely on the MCT method presented in [5], to implement our ICT pipeline. Figure Figure 3.1 represents the input, output and the main steps of this processing pipeline. In these pipeline, first, for each trajectory, the Re-ID features of the vehicle in every frame, i.e., frame-level features, are extracted using a pre-trained ResNet50 network. Then, the frame-level features are combined by weighted averaging, to acquire the trajectory features, i.e., clip-level features. The weights are calculated using an attention generation network. Next, the spatial-temporal constraints are applied on the trajectories, according to the pre-defined trajectory-based camera link models. This step reduces the search space of the next stage, which is trajectory matching. In this stage, the Euclidean distances matrix of the Re-ID features is calculated for every possible trajectory pair, then the trajectories with the smallest distance are greedily matched. To reduce the search space further, we add an optimization step in the matching process, by considering the order of the vehicles in different cameras. We describe the framework in detail in Sections 3.1.1 – 3.1.4.

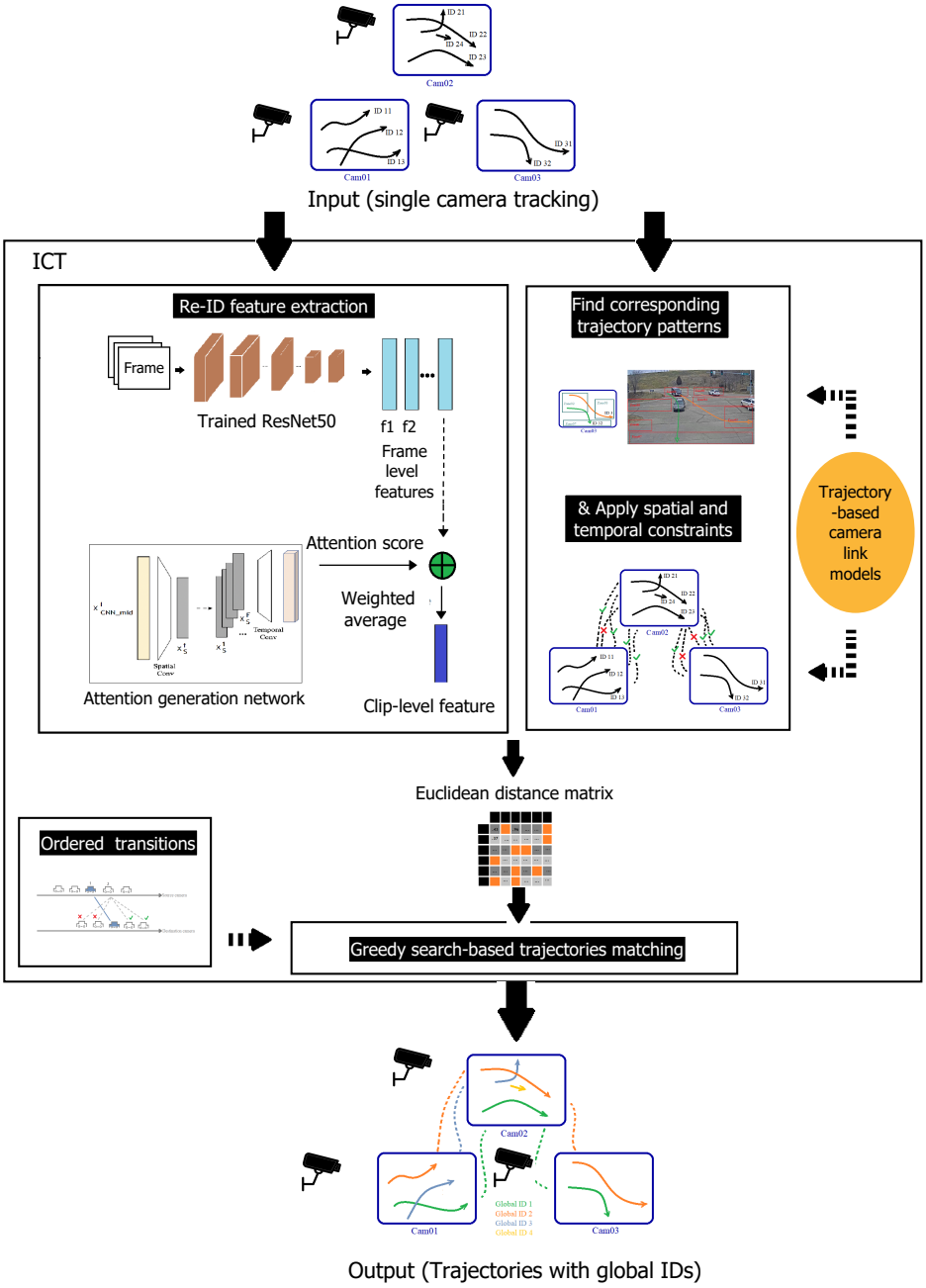


Figure Figure 3.1 The base pipeline of our ICT inspired by [5]. The inputs are the videos recorded by every camera along with the tracking result in each camera (in the figure, each arrow represents the trajectory of a vehicle observed in a camera view. The trajectory of one vehicle in a camera view has a local ID that is only consistent on that camera and varies in other cameras). The input are passed through all the steps of the ICT pipeline to generate the output, which are the complete trajectories of vehicles with unique global IDs across multiple cameras.

We chose to use this framework as our base pipeline for the following reasons:

- State-of-the-art performance on the AIC19 benchmark.
- Open-source.
- Real-time processing speed.
- Adaptable to both roads and parking lots environments.
- Modularity to adapt to future improved ReID models, camera link models, or strategies for considering spatial-temporal constraints.

3.1.1 Input Representation

The inputs of the ICT system are the videos received from the cameras as well as the tracking results in each camera; that is, in every frame a bounding box with a local identifier (ID) indicates the location of every tracked vehicle. A local ID is consistent for the same vehicle in one camera, while it varies in other cameras.

To obtain these tracking results for our ICT system we adopt the TrackletNet Tracker, which is a single camera tracking method proposed in [46].

The TrackletNet tracker is robust in dealing with occlusions and false detections and is based on a *tracklet* graph-based model with three key components: i) *tracklet* (i.e., small incomplete trajectories) generation, ii) connectivity measurement, and iii) graph-based clustering. The *tracklets* are generated based on the intersection-over-union (IOU) of the detection results in each frame, compensated by the appearance similarity between two consecutive frames. Each generated *tracklet* is considered as one node in the graph. The edge weights between nodes shows the likelihood of the two *tracklets* belonging to the same target. The weights are calculated using a multi-scale TrackletNet that is built as a classifier that combines temporal and spatial features in the likelihood estimation. Finally *tracklets* from the same ID are merged into one group using clustering [5]. A diagram of the TrackletNet tracker framework is shown in Figure Figure 3.2.

Given the SCT results, we create the visual trajectories of vehicles in every camera as input. This means, for each vehicle on each camera, we have a sequence of images, i.e. clip, of the vehicle bounding boxes that represents the vehicle movement on that camera.

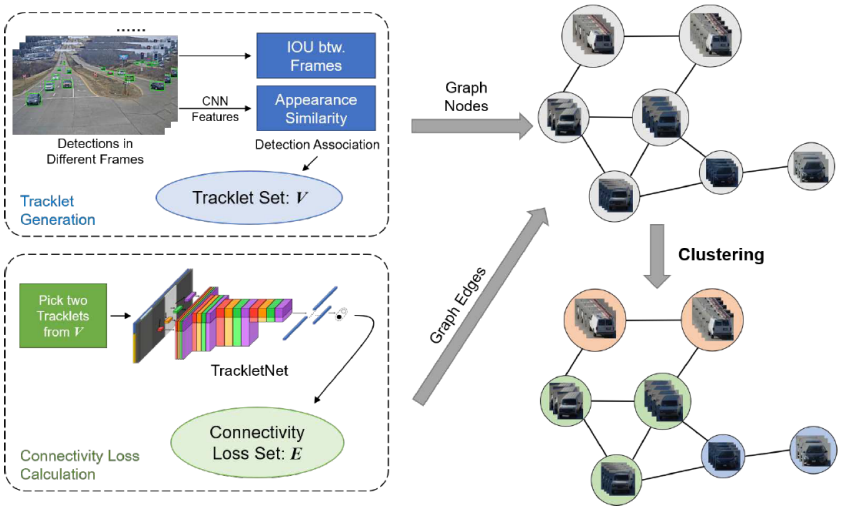


Figure Figure 3.2 The TrackletNet tracker framework used for Single-Camera tracking. Given the detections in every frames, detection association is performed to generate *tracklets*, that form the Vertex Set (V). Next, every two *tracklets* are fed into the TrackletNet to measure the connectivity, that forms the Edge Set (E). A graph model G is created from V and E. Finally, using the graph partition approach,*tracklets* with the same ID are grouped into one cluster [5].

3.1.2 Appearance-based Re-identification

This module creates a feature vector for each trajectory describing the appearance of the related vehicle. These trajectory features are fed to the next module to be used for calculating the distance matrix of trajectories.

Frame-level features: We use a convolutional neural network (CNN) to extract the appearance-based Re-ID features of the tracked vehicles in every frame of a trajectory. This CNN feature extractor uses a ResNet [47] architecture with 50 layers that is pre-trained on ImageNet [48]. The 2048-dimensional fully-connected layer before the classification layer of the network is used to represent the appearance of the vehicle. The architecture of a ResNet50-based feature extractor is shown in Figure Figure 3.3.

Clip-level features: Each trajectory in every camera is composed of consequent frames that are considered as a clip. To represent the appearance features of a trajectory in a single vector, we aggregate the frame-level features of each trajectory and introduce the clip-level features. To do this, an attention generation network proposed in [49] is exploited to calculate attention scores for the frames of the video clips and provide the clip-level features using a weighted average.

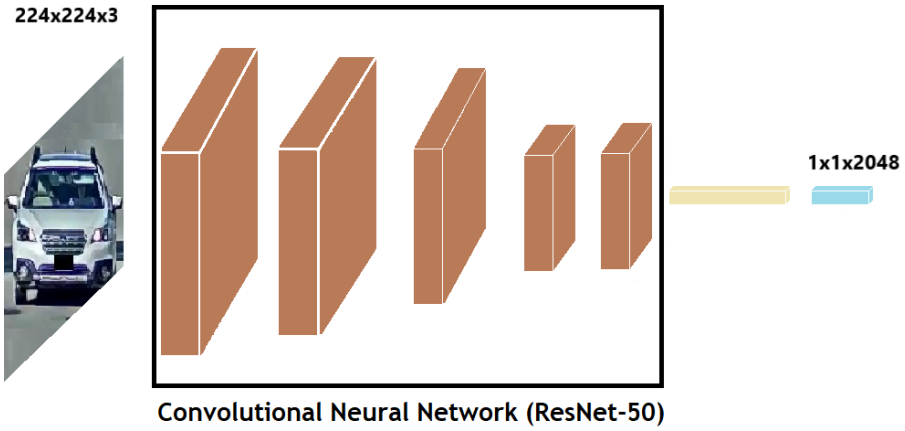


Figure Figure 3.3 Architecture of the ResNet50-based feature extractor used to generate 2048-dimensional vehicle features.

The attention generation network consists of a spatial convolutional network that is a 2D convolution operation and a temporal convolutional network which is a 1D convolution operation (see Figure Figure 3.4). Attention generation mechanism is a weighted average on the sequence of the images features. Given the attention for frame f of clip c as a_{f}, $f \in [1, F]$, then

$$X_{\mathrm{c}} = \frac{1}{F} \sum_{f=1}^{F} a_{\mathrm{f}} X_{\mathrm{f}} \tag{3.1}$$

where F is the number of the frames of the clip, X_{f} is the feature vector of the frame f of the clip c and X_{c} is the feature vector of clip c.

The attention generation network takes a sequence of frame-level features extracted from a ResNet-50 network as inputs, and gives out F attention scores. First, we apply a spatial convolutional layer. The spatial layer brings in focus on the regions of the vehicle image that are more discriminating. Then, we apply the 1D temporal convolutional layer on the output of the spatial convolutional layer to generate temporal attentions s_{f}. The second convolutional layer is called temporal, since it is responsible to find the frames of the vehicle in a clip which are more deterministic of a vehicle appearance, and assign higher scores to those frames accordingly. The final attention scores a_{f} are calculated using Softmax function:

$$a_{\mathrm{f}} = \frac{e^{s_{\mathrm{f}}}}{\sum_{i=1}^{F} e^{s_{\mathrm{i}}}} \tag{3.2}$$

Hsu et al. [5] trained the attention generation network proposed in [49], to get more reliable attention scores for the frames in video clips of vehicles, and we use these trained networks to generate the scores.

The Re-ID model is trained using a combination of cross-entropy loss and *triplet loss*. *Triplet loss* is originally proposed by [50], for face verification, that is similar to vehicle Re-ID task. Given an anchor image, the objective of triplet loss is to minimize the distance between the anchor and a positive image that belongs to the same vehicle, while maximizing the distance between the anchor and a negative image belonging to another vehicle. The data used to train these networks is the Re-ID benchmark of AIC19 dataset[1].

3.1.3 Trajectory-based Camera Link Models and Spatial-temporal Constraints

In every environment (e.g. road, parking lots) there are some traffic rules and structure limitations which restrict vehicle movements. Thanks to these restrictions, we can derive all the possible driving patterns, i.e. trajectory patterns, that can be observed in every camera view. All we need to achieve these patterns are a set of video samples of vehicles traveling in these camera views, and human manual efforts to review these videos, and recognize the patterns. These patterns can be extended across the neighboring cameras. The spatial-temporal relations, and constraints to link these patterns across cameras constitute the trajectory-based camera link models. Here we specifically describe how we define the patterns, and build the camera link models based on these patterns. We also discuss how these models will be used in our ICT to apply temporal and spatial constraints on the trajectory pairs aiming at reducing the candidate pairs for matching.

Trajectory-based camera link models creation: As we discussed earlier, each camera view has a number of trajectory patterns. These trajectory patterns can be linked to only a limited number of patterns in other camera views, due to the spatial limitations of the environment. We maintain all the possible links between trajectory patterns, as well as the transition time between these patterns, and call them the trajectory-based camera link models. To do this, first we divide each field-of-view of cameras into multiple zones. These zones are defined based on the areas such as intersections, the turning points, and the enter/exit points to/from the camera views. Then, we represent the trajectory patterns using these zones. This means, every pattern is uniquely described by an ordered list of zones that a vehicle passes through during its travel. Figure Figure 3.5, shows examples of two trajectories which are defined based on zone lists. The possible links between trajectory patterns are also presented using these zone list.

[1] https://www.aicitychallenge.org/2019-ai-city-challenge/

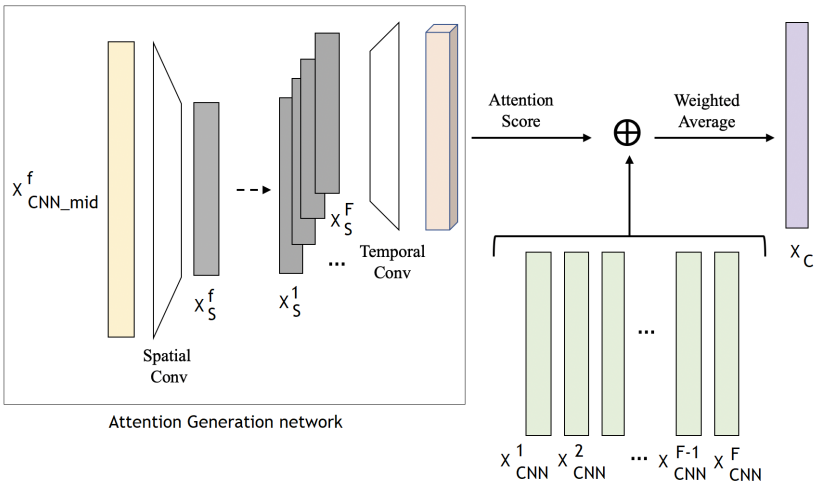


Figure Figure 3.4 Extracting clip-level features using an attention generation model. The frame-level features are passed through the spatial and temporal convolutional networks to obtain the attention score for each frame. Then, the weighted average is calculated using these attention scores to achieve clip-level features [5].

In addition to spatial limitations, we represent temporal relations between the patterns in the camera link models. Particularly, we define transitions between each two neighboring cameras. The transition between a camera pair is defined as $L = (T_{\text{src}}, T_{\text{dst}})$, where $T_{\text{src}} = \{\text{tr}_{\text{src}_1}, \text{tr}_{\text{src}_2}, ..., \text{tr}_{\text{src}_m}\}$ are the trajectories in the source camera and $T_{\text{dst}} = \{\text{tr}_{\text{dst}_1}, \text{tr}_{\text{dst}_2}, ..., \text{tr}_{\text{dst}_n}\}$ are the trajectories in the destination camera. Next, for each transition L, considering all the possible link between the source and destination trajectories and their transition time, we define a time window $(\triangle t_{\min}, \triangle t_{\max})$ indicating the minimum and the maximum transition time that a vehicle can have passing through those two cameras.

Figure Figure 3.6 shows examples of transitions between neighboring cameras. In this figure, different possible transitions between neighboring cameras are shown as trajectories with different colors, meaning these trajectories can be linked. The transition time for each of these possible links is calculated to find the time window.

Spatial Constraints: Given the input trajectories in each camera, we first assign every trajectory into a predefined trajectory pattern, and represent them by their corresponding zone lists. We presume the spatial constraints of matching these trajectories are the same as those of their corresponding patterns which are predefined in camera link models. This way, we apply spatial constraints to find the potentially matchable trajectories across cameras, using our camera link models.

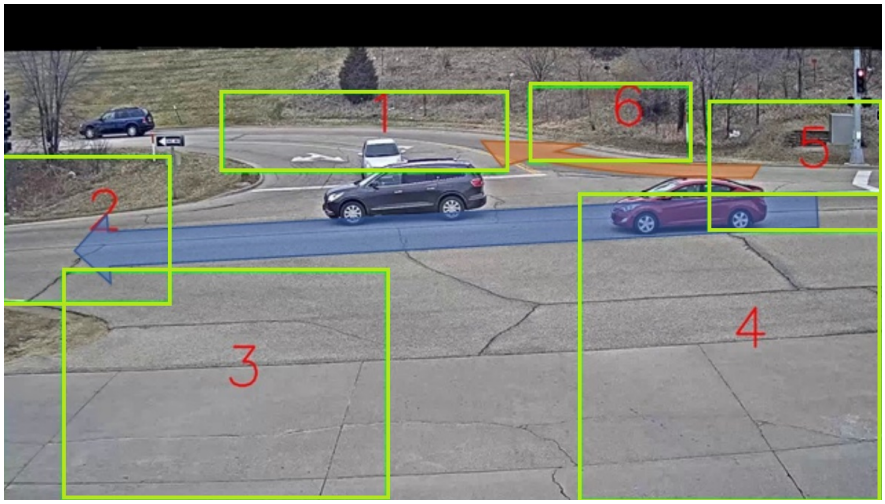


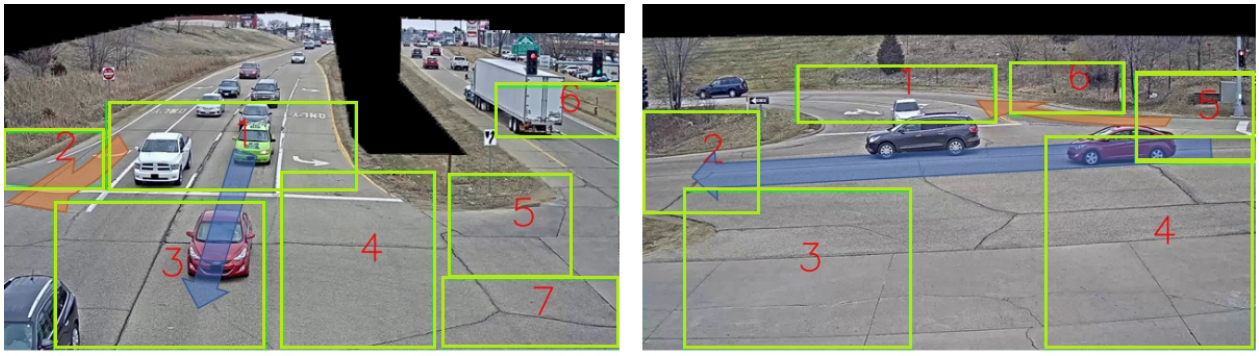
Figure Figure 3.5 Examples of defining trajectories based on zone lists [5]. The trajectory in blue can be described by zone list [5, 2], and the trajectory in orange can be described by zone list [5, 6, 1].

In the process of assigning the trajectories into patterns, the bounding boxes of a tracked vehicle may not go through the corresponding zones perfectly, without touching other zones. This can be due to the viewing angle of the camera or the large sized vehicles. Thus, the trajectory of a tracked vehicle is assigned to the closest existing pattern which is the one that: (i) its zones have the most overlapping area with the actual zones gone through by the tracked vehicle (ii) the order of passing the zones in the pattern and in the actual trajectory are the same.

Temporal Constraints: Given an input pair of trajectories in a destination and source camera, first we find their corresponding trajectory patterns. Then, we utilize temporal relations between the patterns in the camera link models to apply temporal constraints on the corresponding input trajectories for matching, aiming at reducing the number of potentially matchable trajectories. To do this, having the zone lists of each trajectory, we calculate t_{src} and t_{dst}, which are the actual time spots that the vehicle leaves the last zone in the source camera and enters the first zone in the destination camera, respectively. The transition time for this trajectory pair is defined as,

$$\triangle t = t_{\text{src}} - t_{\text{dst}}, \tag{3.3}$$

Then, the trajectory pair is considered as valid and potentially matchable if its transition time is inside the time window of the transition, that is presented in the camera link models.



Two cameras with overlapping views.

Two cameras with non-overlapping views.

Figure Figure 3.6 Examples of transitions between neighboring cameras. The top camera pair have overlapping views, and the bottom camera pair have non-overlapping views. Different possible transitions between cameras are shown as trajectories with different colors. For the top camera pair, from source camera (left) to destination camera (right), transition blue is ([1, 3], [5, 2]), and transition orange is ([1, 2], [5, 6, 1]). For the bottom camera pair, from source camera (left) to destination camera (right), transition blue is ([3, 4], [3, 7], [3, 5], [3, 2, 6], [1, 2], [5, 4, 2]), and transition orange is ([1, 2], [5, 2], [6, 2], [7, 3, 2], [3, 4], [3, 4, 5]) [5].

3.1.4 Trajectory Matching

The final step is to match the trajectories that have similar appearance-based Re-ID features. To do this, some methods solve the problem using algorithms that minimize the distances of all the matched pairs [51], [52]. Whilst, others resort to a greedy iterative scheme due to model complexity as in [53], [54], [55], that finds one pair with the smallest distance at a time [52]. In this piepline, after reducing the potentially matchable trajectories by applying the spatial and transition time constraints, we use the greedy algorithm to match the trajectories.

First, we calculate the distance matrix, including pairwise Euclidean distance of the Re-ID features of all the potentially matchable pairs of trajectories. Following that, we greedily select the smallest pairwise distance to match their trajectories. We repeat the process until there is no candidate pair.

However, matching the trajectories based on solely the pairwise Euclidean distance may result in false matching, because the true trajectory pair may not have always the minimum Euclidean distance from the query trajectory due to variations in illuminations, poses, view points and occlusions. A more reliable distance metric is helpful to reduce the false matching rate.

Ordered transitions: We apply an optimization mechanism to reduce the search space of the Re-ID features further, by considering the order constraints between different tracked vehicles. We assume that in some environments the order of a series of vehicles may not change often due to the area structures or traffic rules. Hence, we define an ordered transition for two tracked vehicles so that the orders of the tracked vehicles in source and destination

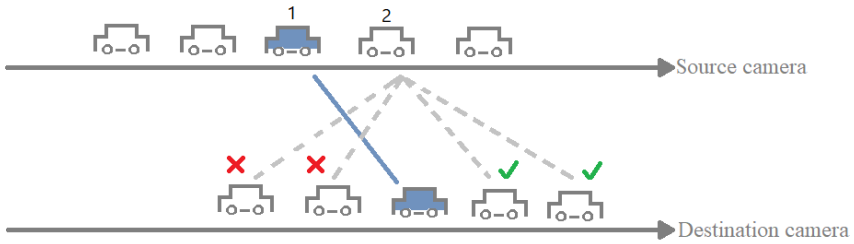


Figure Figure 3.7 Example of ordered transition. When the vehicle pair with the smaller distance is matched in greedy algorithm (blue), the search space of its neighbors is reduced to half because of the order constraint.

camera should be the same. The formalized constraint can be shown as

$$\text{sign}(\text{t}_{\text{src}_1} - \text{t}_{\text{src}_2}) = \text{sign}(\text{t}_{\text{dst}_1} - \text{t}_{\text{dst}_2}), \tag{3.4}$$

Where, t_{src_1} and t_{src_2} are the time of passing the first and second vehicle, respectively within the source camera and t_{dst_1} and t_{dst_2} are the corresponding vehicles passing time within the destination camera. Therefore, while conducting the greedy algorithm, we consider each ordered transition, and remove the potential pairs which conflict with previously matched pairs. Figure Figure 3.7 shows an example of ordered transition, in which we can see that after the vehicle pair with the smaller distance is matched in greedy algorithm, the search space of its neighbor is reduced to half because of the order constraint.

3.2 Effectiveness improvement

To obtain a more efficient and robust ICT system, we revise the pairwise distance measurement of trajectories to deal with intra-class variability and inter-class similarity. Therefore, we take into consideration the k-reciprocal neighbor sets in distance calculation, since it has been proven to be useful for improving the results of the targets association in Re-ID tasks [24]. In addition, we replace the Re-ID model with an ensemble of two modified models, presented in [6], that are pre-trained on a vehicle-based dataset to provide more indicative appearance features for vehicle re-identification.

3.2.1 A robust distance definition based on k-reciprocal nearest neighbors

As we described in Section 3.1.4, we calculate the pairwise Euclidean distance between the appearance Re-ID features of every two trajectories. Then, using a greedy algorithm we match the trajectories with the smallest distance. However, due to variations in illuminations, poses, view points, and occlusions, the true trajectory pair may not always achieve the minimum distance from the query trajectory. Meanwhile, a vehicle with the similar model and appearance may win the contest and mistakenly selected as a match; that is, relying solely on the sample with the smallest distance may result in false matching [24]. To address this issue, when we compare the trajectories, we take into consideration their neighbors as well.

To do so, we exploit the method proposed in [24], that introduces k-reciprocal nearest neighbor sets. We use these sets as contextual knowledge to revise the pairwise distances definition and improve the final target matching results. The idea behind this method is that if two tra-

jectories belong to the same vehicle, it is more likely of them to be appeared in the k-nearest neighbors of each other.

K-reciprocal nearest neighbors: When two trajectories are called k-reciprocal nearest neighbors, it means they are both ranked top-k nearest neighbors when the other trajectory is taken as the query. Therefore, the k-reciprocal nearest neighbor serves as a stricter rule to indicate whether two trajectories are true matches or not, rather than the ordinary k-nearest neighbors [24]. Figure Figure 3.8 shows an example of a query trajectory and its k-reciprocal neighbors set. In this figure, a quarry image and its 7-nearest neighbors are shown on the top row. Only three of these neighbors are considered as reciprocal neighbors, because of emerging the query image in their 7-nearest neighbors. These true matches are shown in green boxes, and the images belonging to the query vehicle are shown in purple boxes.

Jaccard distance: We consider k-reciprocal neighbors sets as contextual knowledge to re-calculate the distance between each pair of trajectories. Following [24], we define a new distance named *Jaccard* distance. The hypothesis behind the proposed distance metric is that if two trajectories are similar, their k-reciprocal nearest neighbor sets overlap and have common samples in their sets. The more common samples, the more similar the two trajectories are. Thus, the *Jaccard* distance between a query trajectory and any other trajectory (e.g. i^{th} sample in the list of all trajectories) can be defined as

$$\text{dist}_{\text{Jaccard}}(\text{Tr}_\text{q}, \text{Tr}_i) = 1 - \frac{|R(\text{Tr}_\text{q}, k) \cap R(\text{Tr}_i, k)|}{|R(\text{Tr}_\text{q}, k) \cup R(\text{Tr}_i, k)|} \tag{3.5}$$

Where $R(\text{Tr}_\text{q}, k)$ and $R(\text{Tr}_i, k)$ represent the k-reciprocal nearest neighbor sets of the query trajectory (Tr_q) and the i^{th} trajectory (Tr_i) of the list, respectively. $|\cdot|$ denotes the number of candidates in the set.

However, to reduce the computational complexity we reformulate the equation as 3.9. First, we encode the neighbor set of each trajectory into an easier but equivalent vector $V_\text{q} = [V_{\text{q},1}, V_{\text{q},2}, ..., V_{\text{q},N}]$. Therefore, V_q denotes the feature vector of the query trajectory which is a N-dimensional vector (N shows the size of the list of trajectories). Each item of the vector indicates whether the corresponding trajectory is included in $R(\text{Tr}_\text{q}, k)$. If the trajectory is not included in the set, the corresponding item will be zero, otherwise it will be filled by the Gaussian kernel of the pairwise distance as

$$V_{\text{q},i} = \begin{cases} e^{-\text{dist}(\text{Tr}_\text{q}, \text{Tr}_i)} & \text{if } i \in R(\text{Tr}_\text{q}, k) \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

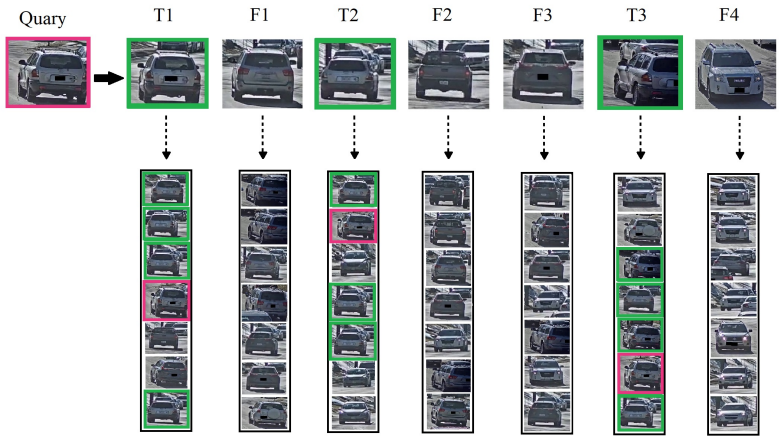


Figure Figure 3.8 Illustration of the reciprocal nearest neighborhoods of a target. Top: The query and its 7-nearest neighbors, where T1-T3 are true matches, F1-F4 are false matches. Bottom: Each column shows 7-nearest neighbors of the corresponding vehicle. Purple and green box correspond to the query and true matches, respectively. The query vehicle and true matches are 7-nearest neighbors reciprocally. Note that each vehicle image here is the representative of a trajectory clip (i.e., a series of images) that for the sake of representation simplicity is shown as one single image.

Given this definition, the number of candidates in the intersection and union set (3.5) can be calculated as

$$|R(\mathrm{Tr}_\mathrm{q}, k) \cap R(\mathrm{Tr}_i, k) = ||\min(V_\mathrm{q}, V_i)||_1 \tag{3.7}$$

$$|R(\mathrm{Tr}_\mathrm{q}, k) \cup R(\mathrm{Tr}_i, k) = ||\max(V_\mathrm{q}, V_i)||_1 \tag{3.8}$$

where min and max operate the element-based minimization and maximization for two input vectors. $||\cdot||_l$ indicates L_1 norm. Finally we can rewrite the *Jaccard* distance as

$$\mathrm{dist}_\mathrm{Jaccard}(\mathrm{Tr}_\mathrm{q}, \mathrm{Tr}_i) = 1 - \frac{\sum_{j=1}^{N} \min(V_{\mathrm{q},j}, V_{i,j})}{\sum_{j=1}^{N} \max(V_{\mathrm{q},j}, V_{i,j})} \tag{3.9}$$

Final hybrid distance: Even though we expand the domain of trajectory comparison to re-calculate their distances, we can not ignore the importance of the original distance (i.e., the euclidean distance between the query trajectory and other trajectories). Therefore, to obtain a robust distance we aggregate the original and the *Jaccard* distance. Thus, the final distance between a query trajectory and each trajectory in the list is calculated as

$$\mathrm{dist}_\mathrm{hybrid}(\mathrm{Tr}_\mathrm{q}, \mathrm{Tr}_i) = (1-\lambda)\,\mathrm{dist}_\mathrm{Jaccard}(\mathrm{Tr}_\mathrm{q}, \mathrm{Tr}_i) + \lambda\,\mathrm{dist}_\mathrm{Euclidean}(\mathrm{Tr}_\mathrm{q}, \mathrm{Tr}_i) \tag{3.10}$$

where $\lambda \in [0, 1]$ denotes the proportion of the original distance ($\text{dist}_{\text{Euclidean}}$) in the final distance calculation. When $\lambda = 1$, only the the original distance is considered. while, if λ= 0, only *Jaccard* distance is considered.

The *Jaccard* distance can be more robust, by extending the k-reciprocal set. In the scenes with extreme variations in illuminations, poses, views and occlusions, the true query pair may not be included in the k-nearest neighbors, and as a result will be excluded from the k-reciprocal nearest neighbors set as well. To address this issue, in addition to taking the k-reciprocal nearest neighbors set of the query trajectory into consideration, we consider the k'-reciprocal nearest neighbors of the candidates in the set, with a condition. Doing this, we add candidates to the set, that are more similar to the candidates in $R(\text{Tr}_\text{q}, k)$ than to the query itself. Therefore, the extended set is defined as:

$$\begin{aligned} R^\star(\text{Tr}_\text{q}, k) &\leftarrow R(\text{Tr}_\text{q}, k) \cup R(\text{Tr}_\text{n}, k') \\ s.t. |R(\text{Tr}_\text{q}, k) \cap R(\text{Tr}_\text{n}, k')| &\geq \frac{2}{3}|R(\text{Tr}_\text{n}, k')|,\ \forall\, \text{Tr}_\text{n} \in R(\text{Tr}_\text{q}, k) \end{aligned} \tag{3.11}$$

The effect of the value of λ, k, and k' is discussed in Chapter 4, Section 4.3.1.

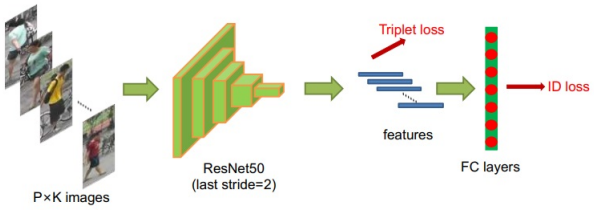
3.2.2 Ensemble re-identification model

In the pipeline introduced in Section 3.1.2, a ResNet50 [47] backbone is used for Re-ID features extraction. We use another public ResNet-based Re-ID model presented by [6], that is modified using some training techniques, to preserve more details of the vehicles, and is trained on the Re-ID set of AIC20 dataset. The architecture of this modified model is shown in Figure Figure 3.9.

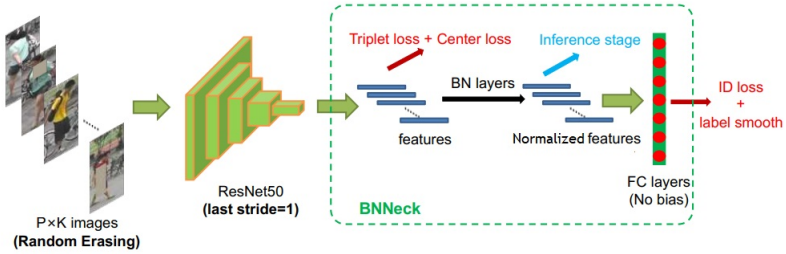
The training techniques are originally proposed by Luo et al. in [56], and are as follows:

- **Warm up learning rate.** Instead of using a large and constant learning rate, a warm up strategy is used to bootstrap the model and improve its performance. To do this, during the first 10 epochs, the learning rate is linearly increased from 3.5×10^-5 to 3.5×10^-4. Then, in the 40^{th} and 70^{th} epoch, the learning rate is set to 3.5×10^-5 and 3.5×10^-6, respectively. The experimental results in [56], show that applying this warm up strategy has a great impact on the model performance.

- **Random erasing augmentation.** A data augmentation method is used to address the occlusion problem. Particularly, in every batch of training images, a number of images are randomly selected, then a random rectangle region of these images are selected and the pixels inside of that region are replaced with random values. This



(a) The pipeline of the standard baseline.



(b) The pipeline of the modified baseline.

Figure Figure 3.9 The pipelines of the standard baseline and the modified baseline. P and K are the number of random vehicle identities, and the number of images per vehicle identity that are sampled to constitute a training batch [56].

way, the occlusion problem is simulated on a group of training images, so the trained model handles the occlusion more effectively.

- **Label smoothing.** As we previously mentioned, cross entropy loss function is used in the baseline Re-ID model training, to train the classification category, i.e., vehicle IDs. Since, the category of the classification is determined by the vehicle ID, this loss function is called ID loss. Label smoothing method proposed in [57], is applied to reformulate ID loss, so that the model is encouraged to be less confident on the truth ID labels of the training set, leading to prevent overfitting.

- **Last stride.** The less spatial down-sampling we have in the model, the higher spatial resolution is obtained. Higher spatial resolution enriches the granularity of the features, and brings improvement in Re-ID. Therefore, in the modified Re-ID model, the last stride, which is considered as a down-sampling operation, is decreased and set to 1, instead of 2.

- **BNNeck.** Most works combine ID loss and triplet loss together to train Re-ID models, in which, ID loss and triplet loss constrain the same feature vector. While, the targets of these two losses are inconsistent in the embedding space. More specifically, ID loss basically optimizes the cosine distances, while triplet loss focuses on the Euclidean dis-

tances. Therefore, using these two losses to simultaneously optimize a feature vector, may result in inconsistency of their goals, for instance, one loss is reduced, while the other loss is increased. To address this problem, a structure is added to the model, named BNNeck. BNNeck adds a batch normalization (BN) layer after features, and before classifier fully-connected layers. In the training stage, the normalized features are used to compute ID loss and the features before the BNNeck are used for computing the triplet loss. In the inference stage, the normalized features are used.
Normalization balances each dimension of the features, so features are normally distributed near the surface of the hyper-sphere in the embedding space. This distribution makes the ID loss easier to converge. Furthermore, BNNeck reduces the constraint of the ID loss on the features, making triplet loss easier to converge simultaneously. Experimental results in [56], show that BNNeck substantially improves performance of the Re-ID model.

- **Center loss.** The baseline Re-ID model is trained using triplet loss and ID loss. Triplet loss ignores the absolute values of the feature distances of positive pairs and negative pairs, and only considers the difference between them. To compensate for this problem, center loss [58] is used in the modified Re-ID model. Center loss learns a center for features of each class (ID), and penalizes the distances between the features and their corresponding class centers. Thus, the modified Re-ID model contains three losses, including ID loss, triplet loss, and center loss.

Studies show that an aggregation of Re-ID models with different backbone enhances the task accuracy [6], because some models with specific architectures and settings focus on details, while others consider the global features of the objects. Thus, a combination of features extracted from such models tends to provide more variant information that strengthens re-identification. Therefore, we tried aggregation of multiple backbones with different architecture (ResNet-50, ResNet-101, and ResNet152) to provide varying levels of detail. The effects of different backbone networks are presented in Section 4.3.1. The final Re-ID model we chose based on the best performance we achieved on the experiments described in Chapter 4, is an ensemble of ResNet-101 and ResNet-50 [6] backbone networks.

We present our final ICT pipeline using the ensemble Re-ID model and the robust hybrid distance definition in Figure Figure 3.10.

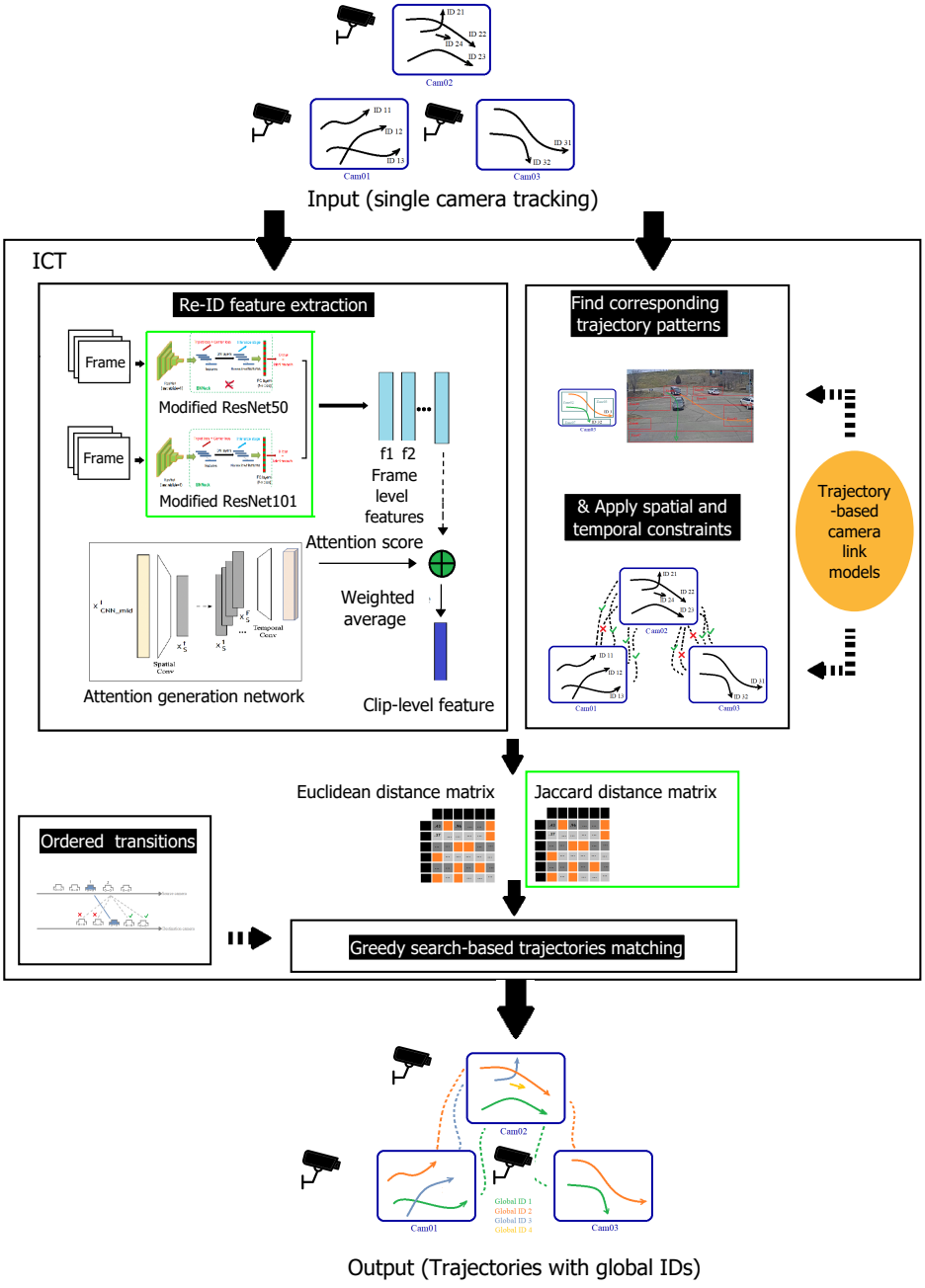


Figure Figure 3.10 Our proposed ICT pipeline. The modifications compared to the base pipeline are shown in green boxes. We replace the ResNet50 Re-ID model by an ensemble of two modified Re-ID models to achieve more deterministic Re-ID features. We also used a robust hybrid distance metric, composed of Euclidean distance and Jaccard distance, which is calculated based on k-reciprocal nearest neighbors, to deal with inter-class similarity and intra-class variability challenges.

3.3 Resources usage improvement

In this work we aim to develop an end-to-end ICT that performs in real-time. A real-time system maintains more than one frame per second (FPS) processing speed. Our ICT delivers speed of 12 FPS. Although it reports a high speed, this speed is decreased when there are more cameras and/or more crowded scenes. Hence, we are interested in optimizing the implemented code to present an ICT that remains efficient in difficult conditions. Besides, reducing processing execution overhead is one of the ways on enabling ICTs employment on edge devices.

Furthermore, Memory usage is one of the biggest barriers in running existing ICTs when we have large datasets. Since the ICTs are usually run on computers with quite high resource capacities, they are not implemented efficiently in terms of memory usages. Thus, they usually cannot be run under memory limited resources. However, powerful computers may not be always available and this motivates us to design an efficient ICT system in terms of memory consumption.

Thus, in order to present an efficient implementation of the ICT, we present a methodology to identify our proposed ICT bottlenecks and indicate points of improvement in terms of execution time and memory consumption. Memory provides the processor space to manipulate data and run programs. While, execution time represents the time spent by the system to perform the tasks. We aim to propose an implementation with low processing time that is enabled to execute on memory-limited systems. We have formalized the required steps to improve these two factors in Sections 3.3.1 and 3.3.2.

3.3.1 Execution time

Here we present a methodology to analyze, identify and reduce time execution overhead of ICT systems. Our methodology consists of three main operations, represented as three blocks in Figure Figure 3.11: i) *hot spots detection* ii) *improvement* iii) *evaluation*.

Hot spots detection: This step consists of detection of the most time-consuming part of the code, by measuring and comparing the execution time of each module/function/line of the code. Monitoring the execution time in the module and function level is useful to recognize the low-performance implementation of the algorithms, which need efforts for re-implementation. It is also essential to extend the monitoring up to code-line level, because it is helpful to detect the inefficient instructions, data structures, etc. that are often easily replaceable. To do this, according to the programming language of our implemented code,

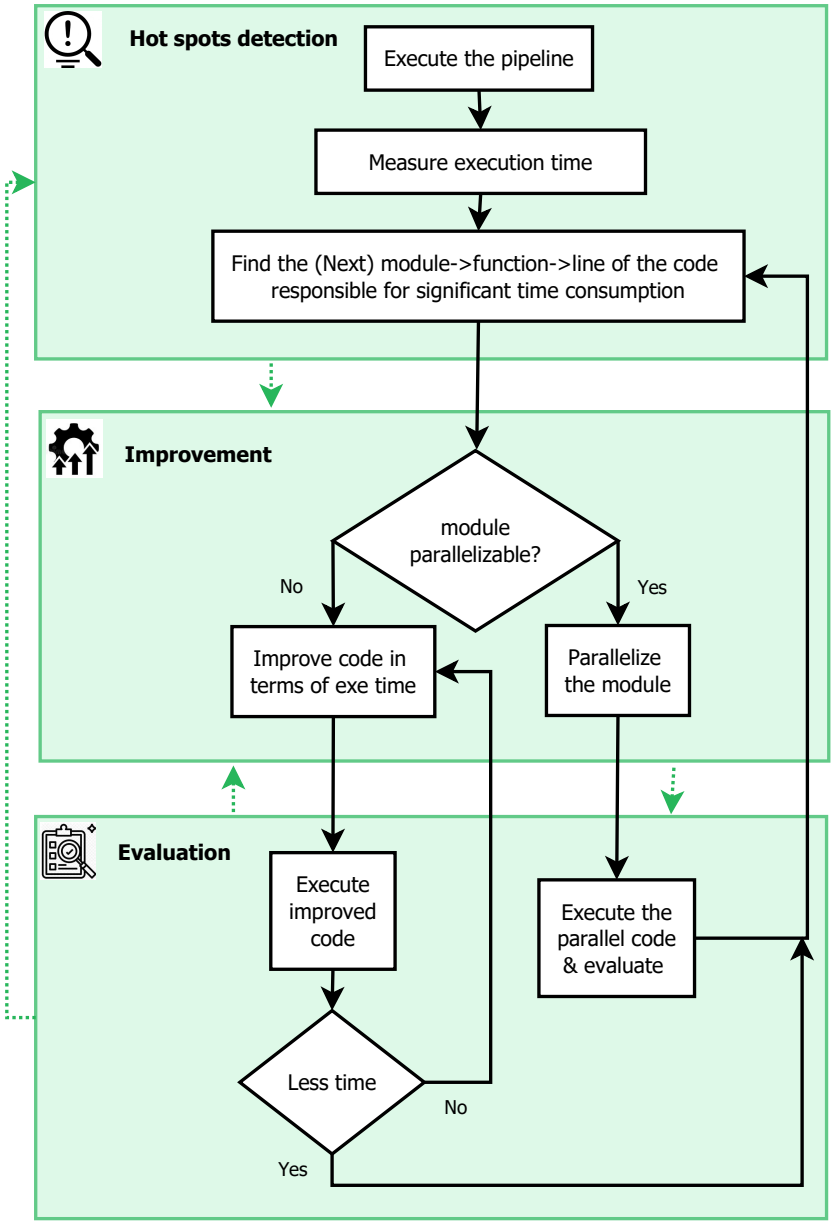


Figure Figure 3.11 Proposed workflow to optimize the execution time of an implemented system. The green blocks represent the main steps (detection, optimization and evaluation), and the green dotted arrows show the directed connections among these blocks. The shapes and arrows in black illustrate the steps in detail.

we can use any time tracking packages and tools (e.g. *datetime* [2], *Profile* [3] or Scalene [4] in Python).

Improvement: In this step, we tackle the identified CPU time usage hot spots to find a solution to speed them up. Depending on where the execution time is spent, the solution varies from providing an optimal controller sequence all the way to re-implement the algorithm from scratch. One of the most promising solutions to improve the CPU usages, and reduce execution time substantially is parallelization. Thus, prior to any actions of improvement, we analyze the implemented modules to check their capability for parallelization, and re-implement them in parallel if it is applicable. Then, we perform line-code level improvement actions on that module.

Evaluation: In the this stage, we evaluate the impact of the improvement achieved in the previous step, by re-executing the code and comparing the new execution time with the previous one. Having been ensured that we achieve the same results in terms of our evaluation metrics, if the measure resulted in faster we proceed the cycle (*hot spots detection*, *improvement*, *evaluation*), otherwise we try other solutions toward optimizing that part. We loop till we meet required criteria. This is done for each possible point of improvement.

3.3.2 Memory Consumption

Similarly to the execution time improvement methodology, we have established a new methodology to reduce the memory footprint of the ICT application.

Figure Figure 3.12 shows our proposed methodology. It consists of three main operations, represented as three blocks: i) *hot spots detection*, ii) *improvement* iii) *evaluation*.

Hot spots detection: This stage is responsible for detecting the most memory demanding parts of the code by measuring and comparing the memory usage of each module/function/line of the code. We also monitor the memory consumption for the action of copying over time, to discover unreasonable trends, that may reveal some memory performance problems. In addition, we separate out memory consumed by the native code and developed code:

- **Native code**, mostly includes either the programming language implementation (in our application Python) or external libraries.
- **Developed code**, refers to the code developed by the programmers.

[2] `https://docs.python.org/3/library/datetime.html`
[3] `https://docs.python.org/3/library/profile.html`
[4] `https://github.com/plasma-umass/scalene`

We provide these information and measurements using available profiling tools, depending on the programming language of the implemented code.

Improvement: In this step, depending on the source of the high memory usage, i.e., native code or developed code, we take the appropriate approach to improve the implementation: if the developed code is accountable for the massive memory consumption, we focus on improving our developed code. The improvements includes using appropriate data structures with low-memory consumption, avoiding redundant making copies of the data, as well as memory leakage, etc. Otherwise, if the memory is consumed by native code, we assume there is a performance problem in an external library we are using. A solution to tackle this problem is to replace the libraries or using them more efficiently. Also, a possible solution is to write proprietary code to improve the performance.

Evaluation: In this step, we evaluate the effectiveness of the improvement introduced in the previous step. If the measure brought in lower memory consumption we proceed the cycle (*hot spots detection*, *improvement*, *evaluation*), otherwise we try other solutions for improving that part of the code. We loop till we meet required criteria. This is done for each possible point of improvement.

3.4 Summary

In summary, in this chapter we:

- introduced the base pipeline that we selected to design our ICT inspiring by, and explained its components in detail.
- modified the base pipeline and improved its performance from two aspects of
  - **effectiveness** by
    (1) replacing the ResNet50 Re-ID model by an ensemble of two modified Re-ID models to achieve more deterministic Re-ID features.
    (2) using a robust hybrid distance metric, composed of Euclidean distance and Jaccard distance, which is calculated based on k-reciprocal nearest neighbors, to deal with inter-class similarity and intra-class variability challenges.
  - **resource usage** by proposing
    (1) a methodology to analyze, and reduce execution time of ICTs.
    (2) a methodology to analyze and reduce memory footprint of ICTs.

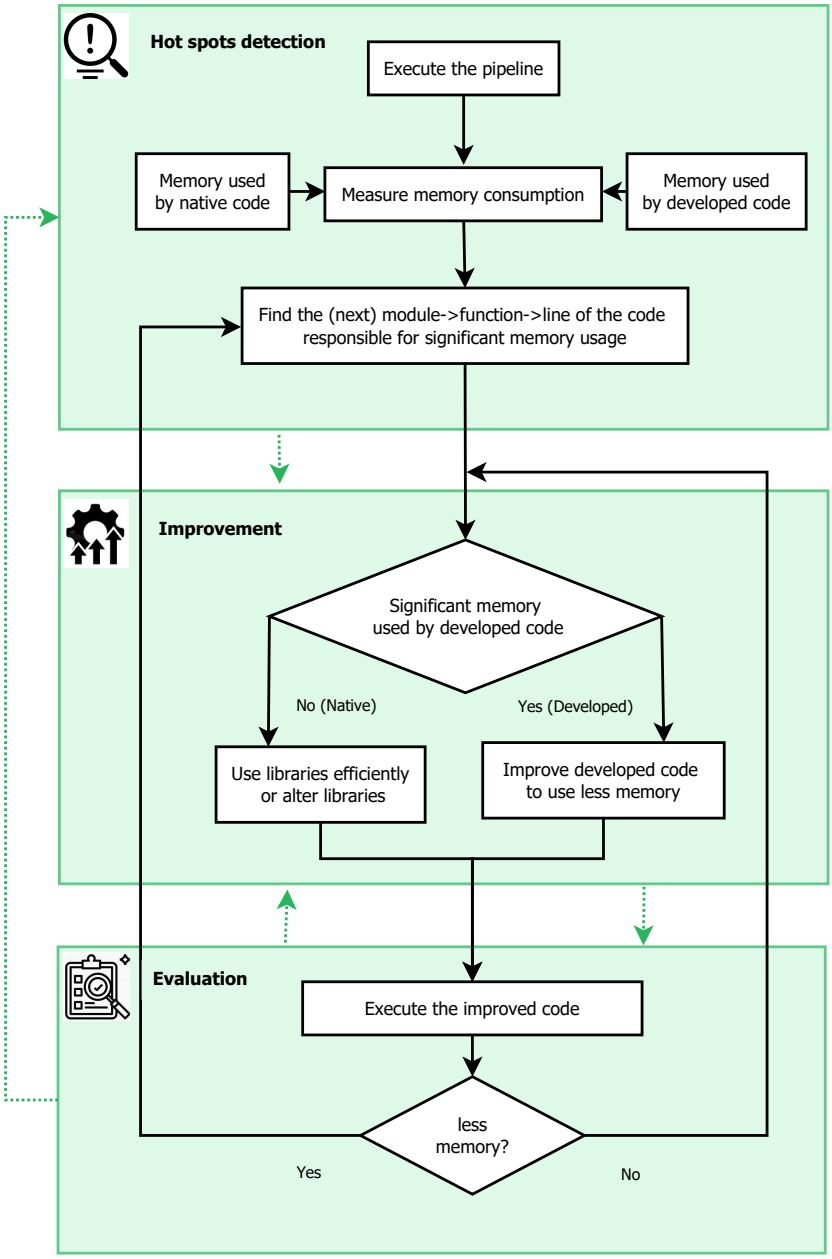


Figure Figure 3.12 Proposed workflow to optimize the memory consumption of an implemented system. The green blocks represent the main steps (detection, optimization and evaluation), and the green dotted arrows show the directed connections among these blocks. The shapes and arrows in black illustrate the steps in detail.

CHAPTER 4 IMPLEMENTATION AND RESULTS

In this chapter, we describe all the details about the implementation of our method. First, we review the dataset and benchmark we used for the majority of our experiments and evaluations, and also the data provided by our industrial partner, Cysca Technologies, that are used for visual evaluations. In addition, we introduce our proposed synthetic dataset with challenging weather conditions. Then, we describe the metrics to measure the performance of our ICT. Next, we provide information about the testing environments, and in the last two sections we present our experimental results and summarize our findings.

4.1 Dataset and benchmark for evaluation

We used the AIC20 dataset [59] from AI CITY challenge 2020[1] for most of our experiments, visualizations, and evaluation. We created a synthetic dataset containing snow, rain, and windy weather in the scenes, based on AIC20 benchmark to validate our method in difficult weather conditions. We also used video sequences recorded from Cysca private parking lot to visually validate our method in a diverse environment with a different camera network configurations and field of views. More details about these datasets are presented in Sections 4.1.1 - 4.1.3.

4.1.1 AIC20

AIC database is an annotated dataset that has been provided annually by the holder community of AI CITY Challenges for the participants since 2017. These datasets are one of the large and comprehensive data that are adopted for the task of vehicle Re-ID and multi-camera tracking.

The 4$^{\text{th}}$ edition of the dataset is referred to as AIC20. Data for this set comes from multiple traffic cameras from a city in the United States as well as the Iowa state highways [59]. Specifically, there are time-synchronized video which are fed from multiple cameras, and are installed in major travel arteries of the city. Most of these videos are high resolution 1080p captured at 10 frames per second. The AIC20 dataset includes CityFlow benchmark [15], [60], adopted for Re-ID as well as multi-camera tracking task, VehicleX benchmark [61], [62], adopted for Re-ID task, and Iowa DOT [63] dataset for anomaly event detection and vehicle counting task. Thus, we used the annotated CityFlow-based subset of AIC20 to evaluate our

[1] https://www.aicitychallenge.org/2020-ai-city-challenge/

ICT method. The annotations of the data present the following attributes:

- Bounding box coordinates,
- Camera ID,
- Vehicle ID,
- Frame ID,
- Latitude,
- Longitude.

The dataset is captured by 46 cameras containing a total of nearly 300K bounding boxes for nearly 900 annotated vehicle identities, divided into 6 scenarios. Three of the scenarios are used for training, two scenarios are for validation and the other scenario is for testing. There are 215.03 minutes of videos in total. Given that the evaluation with the test set of AIC20 requires a high memory resource, we used the validation set of the data set, including two scenarios (scenario no.2 and scenario no.5), for the evaluation of the pipeline. We selected the validation set over the test set, because it is larger, and comprehensive enough to include various scenes with both overlapping and non-overlapping cameras, while in the test set, there are only videos from intersections. The reason we chose the validation set over the training set, is its smaller footprint. For the comparisons experiments we narrowed down the data set and considered only the scenario no.2 of the validation set with 4 cameras (footage time of 14m30s). These videos have frame rates of 10 FPS with frame size of 1080×1920. Figure Figure 4.1 presents the sample frames of six cameras in AIC20 benchmark, and their geolocations.

4.1.2 Videos with difficult weather conditions

In real-world monitoring, difficult weather conditions (e.g. snowy, rainy, windy, etc.) affect the vehicle tracking and re-identification performance [64]. The AIC20 dataset does not include videos with challenging weather conditions. Therefore, the performance of the state-of-the-art trackers proposed in the annual AI CITY challenges are not evaluated in difficult weather conditions. An annotated dataset from the same environment but with bad weather condition is required to evaluate an ICT performance in bad weather, and to compare it with the performance in normal weather conditions. Hence, we created a synthetic dataset based on the AIC20 data. Specifically, we added synthetic snow, rain and wind effects on the

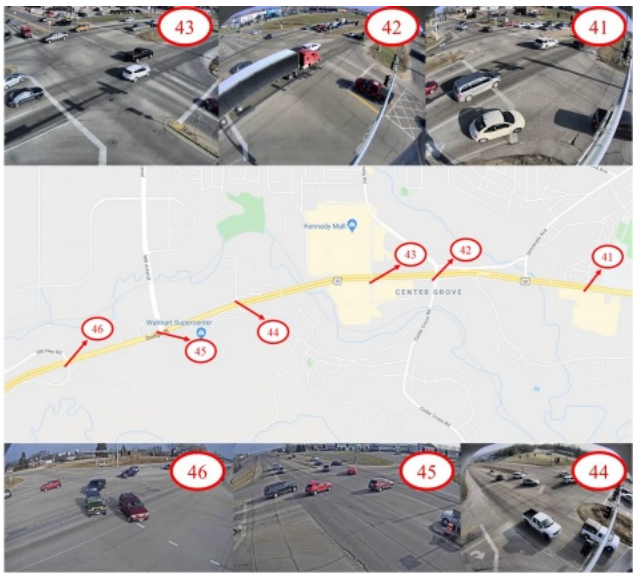


Figure Figure 4.1 Example frames of six cameras and their geolocations [59] from the CityFlow data [15] included in AIC20 benchmark.

existing annotated data, using Adobe After Effects software [2]. The software provides flexible tools so that we can set various properties of rain and snow (e.g. wind, light effect, birth rate, flake or streak size, etc.). Figure Figure 4.2 illustrates example frames of this synthetic dataset.

4.1.3 Cysca's test videos

We used sample videos recorded from Cysca private parking lot in Repentigny. The videos cannot be used for quantitative evaluations because they are not annotated. Therefore, these videos are adopted to visually evaluate our method in a parking lot that has different structure compared to the highways and streets in the AIC20 data. We also validate the application of our pipeline in a situation of having a new setting of cameras in a new environment. In the parking lot there are two cameras installed with overlapping fields of views, providing complementary coverage of the parking area (Figure Figure 4.3). The two cameras are referred to as the west camera and north camera. The videos are recorded at 15 FPS with a resolution of 1280×720 pixels.

[2] https://www.adobe.com/ca/products/aftereffects.html

Figure Figure 4.2 Example frames of the created synthetic dataset based on AIC20 benchmark, containing snow, rain and wind effects.

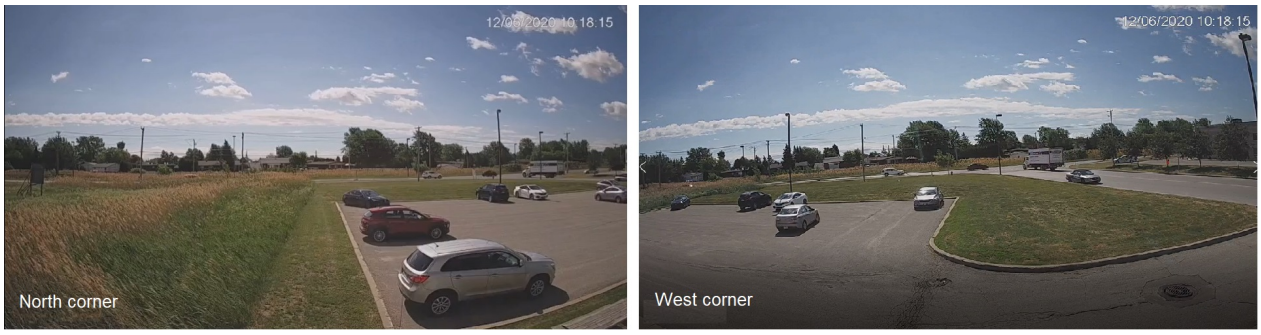


Figure Figure 4.3 Sample frames from the videos of the north and west cameras installed in Cysca parking lot

4.1.4 Evaluation metrics

We adopt the IDentification F1 score (IDF1), which is the main metric to evaluate the state-of-the-art MCT systems [60]. IDF1 score is originally presented in [65]. Ristani. et al. in [65] propose to measure the tracker performance by how long the tracker correctly identifies targets. The measurement occurs in two steps:

- Ground-truth identities are first matched to the identities computed by the tracker. This process is called Truth-To-Result Match. More specifically, each ground-truth trajectory is associated to exactly one computed trajectory by minimizing the number of mismatched frames over all the available data. A mismatch is either a fragmentation or a merge. A fragmentation occurs in a frame if the tracker switches the identity of a trajectory in that frame, while the corresponding ground-truth identity does not change. A merge, which is the reverse of a fragmentation, occurs when the tracker merges two different ground truth identities into one between two frames [65].
- The standard measures of IDentification Precision (IDP), IDentification Recall (IDR), and IDF1 are then calculated on top of this truth-to-result match, by measuring the number of mismatched or unmatched detection-frames, regardless of where the discrepancies start or end or which cameras are involved [65].

IDP is the fraction of computed detections that are correctly identified, while IDR is the fraction of ground truth that are correctly identified. IDF1 balances these two metrics, and measures the ratio of correctly identified detections over the average number of ground-truth and computed detections. A high IDF1 score is obtained when the correct multi-camera vehicles were discovered, accurately tracked within each video, and labeled with a consistent ID across all videos in the dataset [59].

IDF1 score uses the following measures: ID match True Positives (IDTP), ID match False Positives (IDFP) and ID match False Negatives (IDFN) [65]. The IDTP metric measures how many vehicles are detected and truly identified, while the IDFP measures the number of identified vehicles that are not in the ground truth and the IDFN measures how many vehicles of the ground truth are missed among the identified vehicles. The IDF1 score is defined as follows:

$$\text{IDF1} = \frac{2 \times \text{IDTP}}{2 \times \text{IDTP} + \text{IDFP} + \text{IDFN}} \tag{4.1}$$

and the IDP and IDR are defined as follows:

$$\text{IDP} = \frac{\text{IDTP}}{\text{IDTP} + \text{IDFP}} \tag{4.2}$$

$$\text{IDR} = \frac{\text{IDTP}}{\text{IDTP} + \text{IDFN}} \tag{4.3}$$

We used the evaluation script that is provided with the data in the AIC20 benchmark to evaluate our ICT effectiveness. This script takes the ground truth and the results computed by the tracker, and calculates the three metrics of IDF1, IDP, and IDR.

4.2 Implementation and environment

We used Python as our programming language of choice to implement our pipeline due to the large number of open source libraries and packages in Python available in the field of computer vision and deep learning. Python is the most commonly used programming language in computer vision, and it has a large community. This means, there are a lot of online resources and documents regarding problems one may face while developing their code in Python. Our validation architecture to run the ICT and perform the experiments is an AMD Ryzen 7 machine equipped with a RTX 6000 GPU.

4.3 Results and discussion

Here we first present the results of our proposed pipeline and compare its effectiveness to the baseline pipeline in terms of IDF1, IDP and IDR metrics. We also show samples of visual results of the proposed ICT on AIC20 videos. Then, we show the results of the resources usage improvement strategies on the proposed ICT. Some samples of qualitative results of the proposed ICT on the Cysca's parking lot videos are then presented, and finally we report the performance of our ICT in difficult weather conditions.

4.3.1 Effectiveness improvement results

We replaced the conventional Euclidean pairwise distance with a hybrid distance in the ICT, to address the issue of intra-class variability and inter-class similarity in target re-identification and matching. We also adopt an ensemble of two modified backbones to serve as a Re-ID feature extractor to obtain a more performant ICT. The effects of each modification on the ICT performance are indicated in Table Table 4.1. In this table, the first row shows the Baseline ICT performance in terms of IDF1, IDP and IDR metrics. The second and third rows reveal the performance improvement when hybrid distance, and the ensemble Re-ID model are used in the baseline ICT, respectively. The last row indicates the performance of our ICT, including both hybrid distance, and the ensemble Re-ID model.

Table Table 4.1: The impact of each modification on the ICT performance. The last row shows the evaluation results of our proposed ICT. The evaluations are performed on the validation set of the AIC20 dataset(scenario no.2 and scenario no.5).

Method	IDF1(%)	IDP(%)	IDR(%)
Baseline ICT	70.1	69.4	70.8
+Hybrid distance	71.8	69.6	73.9
+Ensemble Re-ID	71.3	70.5	72.1
Our ICT	**72.1**	**69.7**	**74.7**

Table Table 4.2: The effect of k and k' values on the ICT performance in terms of IDF1. In our experiments the λ is set to 0.5. The evaluations are performed on the scenario no.2 of the validation set of the AIC20 dataset.

k	k'	IDF1(%)
30	25	81.3
30	20	81.4
30	15	79.9
25	20	**82.4**
25	15	82
25	10	81.8
20	15	81.1
20	10	80.7
20	5	80.7
15	10	79.5
15	5	79.5

Table Table 4.3: The effect of λ values on the ICT performance in terms of IDF1. In our experiments the k and k' are set to 20 and 25, respectively. The evaluations are performed on the scenario no.2 of the validation set of the AIC20 dataset.

λ	IDF1(%)
0	81.3
0.2	80.7
0.3	81.4
0.4	**81.5**
0.5	81.2
0.6	80.3
0.7	80.3
0.8	79.7
0.9	79.9
1	79.2

Table Table 4.4: The effects of different backbone networks and merging of them on the ICT performance in terms of IDF1. The evaluations are performed on the scenario no.2 of the validation set of the AIC20 dataset.

ResNet-50	ResNet-101	ResNet-151	IDF1(%)
✓			81.1
✓	✓		**82.9**
✓		✓	80.9
	✓	✓	80.5
✓	✓	✓	81.4

Regarding the results shown in Table Table 4.1, using the hybrid distance metric enriches the ICT in terms of IDR metric, because the focus of distance measurement strategy is on providing candidates to compare with the query vehicle. The *Jaccard* distance part of the hybrid distance, allows the system to consider other neighbors of each query in addition to the nearest neighbor; therefore, it increases the chance of finding and matching the right query pair, while it may not have the shortest Euclidean distance to the query due to the intra-class variability or inter-class similarity challenges. Therefore, we can see a noticeable enhancement in IDR metric and a slight increase on IDP as well.

As for the Re-ID feature extractor, since it changes the features that represent each target, it can have impact on both precision and recall of the system, so as it is shown in Table Table 4.1, it has almost the same impact on the IDP and IDR metrics.

By applying both modifications to the baseline pipeline, we achieve an increase on all the metrics compared to the baseline and also obtain the highest rank of IDF1 score that represents a balance of both IDP and IDR metrics.

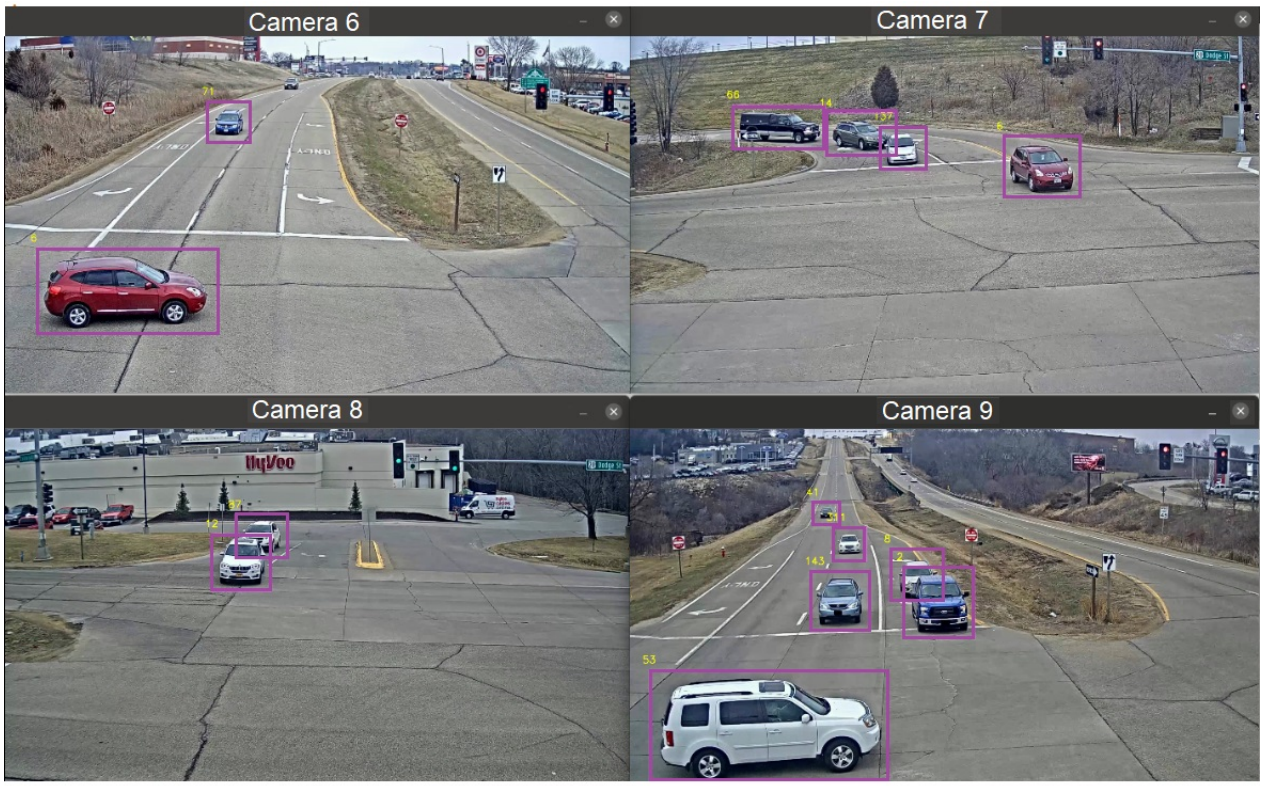
Table Table 4.2 and Table 4.3 shows the results of the experiments we performed to see the effects of k, k', and λ parameters, to find the best configuration for the hybrid distance measurement. The effects of different backbone networks are also investigated and indicated in Table Table 4.4. In these experiments we run the code on only the scenario no.2 of the validation set, described in Section 4.1, to discourage excessive dependency of the parameters' values to the data. In Table Table 4.2, in each row, we can see the IDF1 score of our ICT when k, and k' parameters are set to the specified values for *Jaccard* distance measurement. Similarly, in Table Table 4.3, each row illustrates the IDF1 score of our ICT when λ parameter is set to the specified value. When $\lambda = 1$, only the the original distance is considered, and if $\lambda = 0$, only *Jaccard* distance is considered. In Table Table 4.4, the backbone(s) that are included in the ensemble model are checkmarked in each row, and the last column shows the IDF1 score of our ICT when this combination of backbones are used. According to the

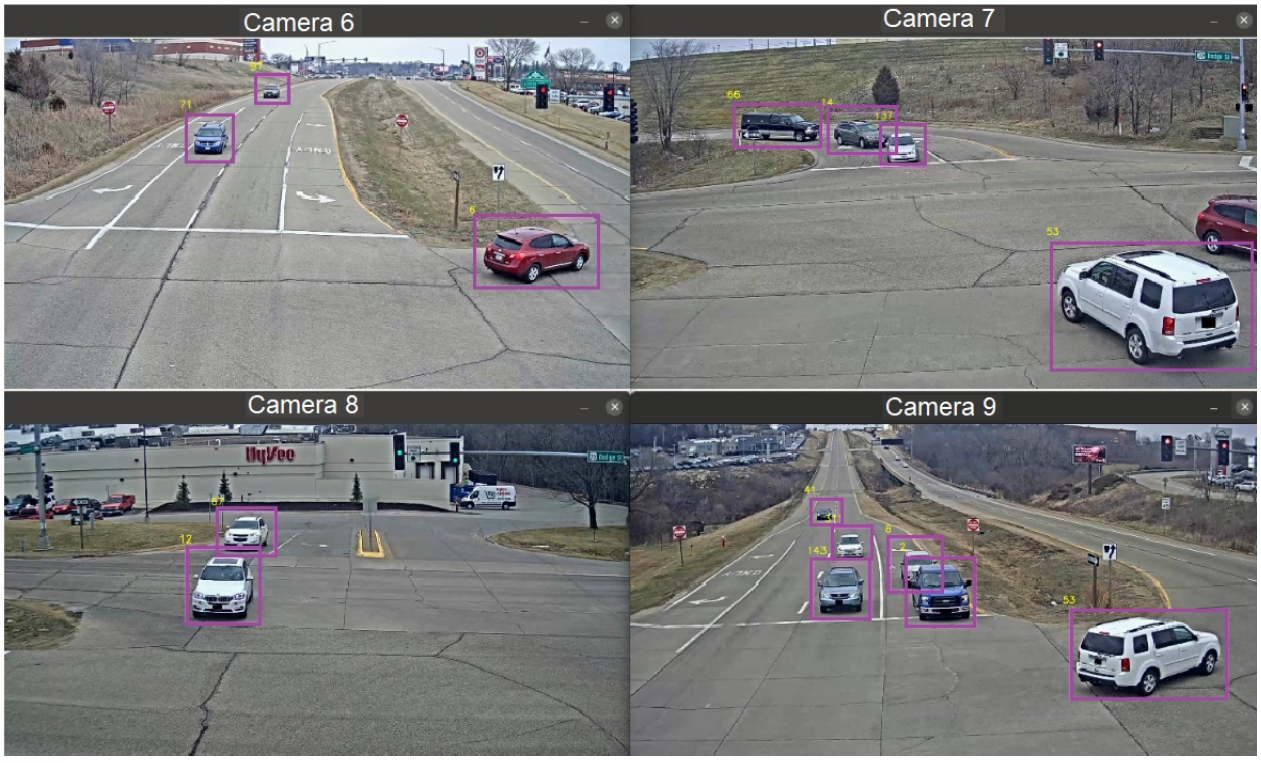
experiment results, the best performance are achieved when the parameters are set as $k = 25$, $k' = 20$, $\lambda = 0.4$, and use the ensemble of ResNet-101 and ResNet-50 networks for Re-ID features extraction.

Visual results on AIC20

We illustrate the tracking results on the videos from AIC20 dataset, to qualitatively evaluate our proposed ICT. Figure Figure 4.4 shows samples of subsequent frames captured from four cameras with overlapping and non-overlapping views. In this figure, there are vehicles assigned to their global IDs that are consistent in different cameras, even though there are appearance variation due to the view angle changes. In Figure Figure 4.5 there are another sample results containing vehicles with the similar appearance and color that are tracked correctly by our trackers even though two of them are moving close to each other, because our ICT does not solely depends on the appearance features, and it considers the spatial-temporal constraints imposed by the camera-link models. Figure Figure 4.6 represents a sample of our ICT failure in matching the same vehicle in two cameras due to the miss detection in one of the cameras.

Figure Figure 4.4 Sample results on scenario no.2 of AIC20, including 4 cameras. The two scenes are 50 frames apart. In the first scene, the vehicle#0 in green color is seen in three cameras (camera-6, camera-7, camera-8) and has the same global ID. In the second scene this vehicle is appeared in camera-9 with the same ID, while it keeps its ID in camera-7.


Camera 6
Camera 7
Camera 8
Camera 9
HyVee


Camera 6
Camera 7
Camera 8
Camera 9
HyVee

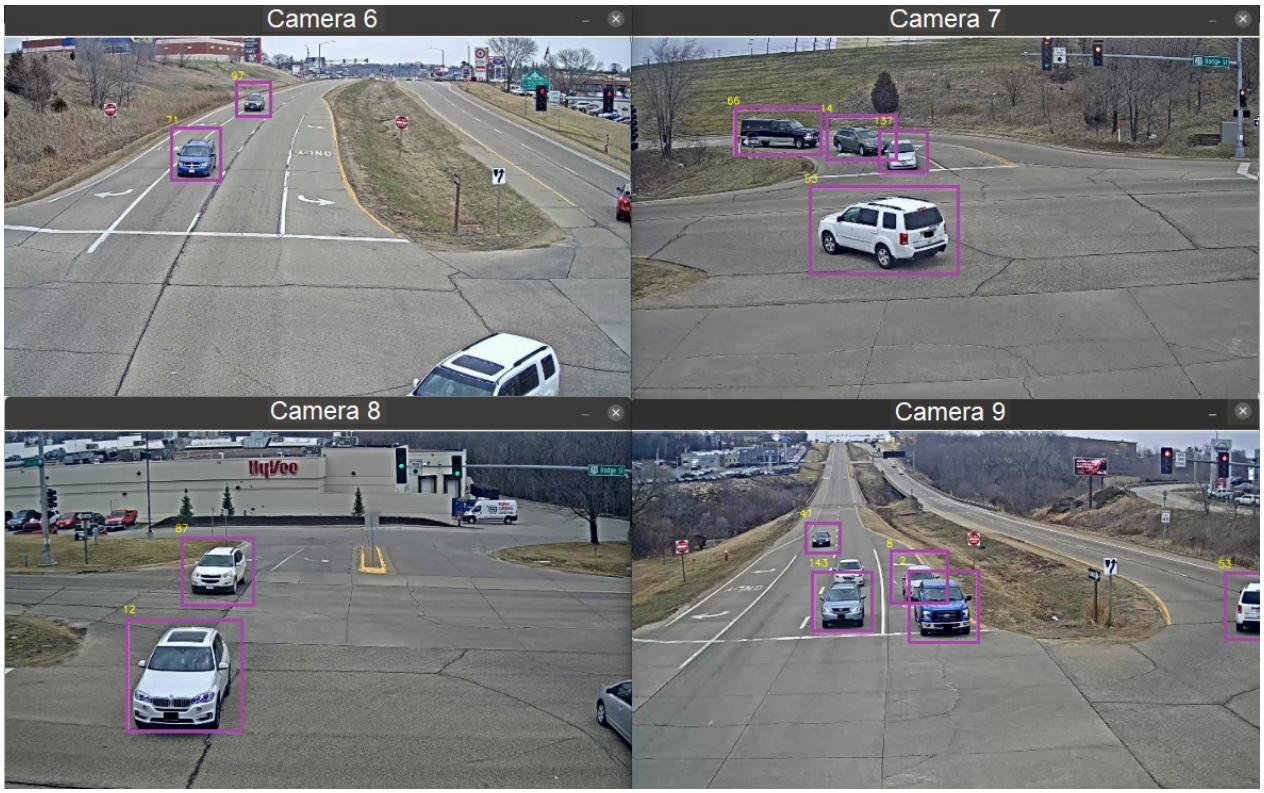


Figure Figure 4.5 Sample results on scenario no.2 of AIC20, including 4 cameras. The three scenes are captured in about 70 frame span. In the first scene, there are vehicles no.12, no.87 in camera no.8 and vehicle no.53 in camera no.7 and camera no.9 with similar look and all in white color and they travel through the same direction. Our ICT tracks and keeps their global IDS during their travel.

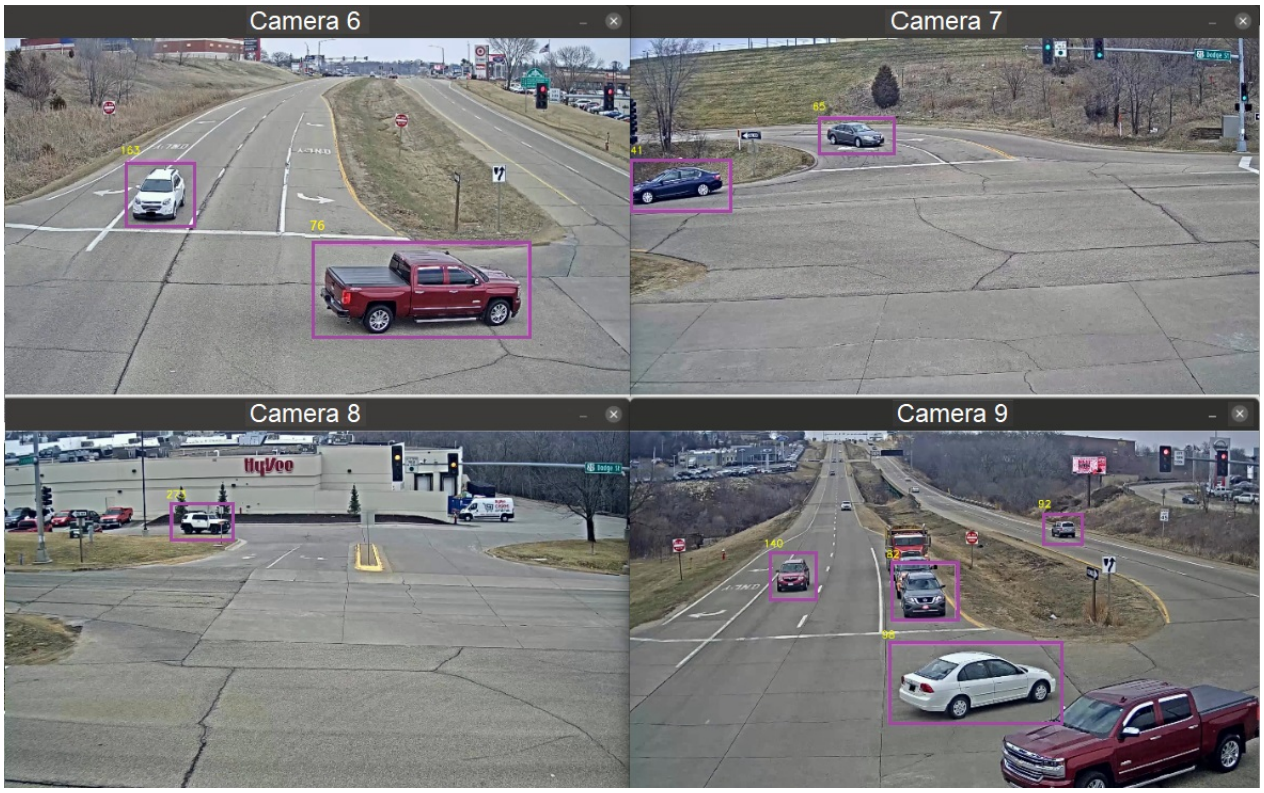


Figure Figure 4.6 Sample of ICT failure in matching the same vehicle. The vehicle#76 in camera-6 is appeared in camera-9 as well, however it is not detected in this camera, subsequently the global ID of 76 is not assigned to the vehicle.

4.3.2 Resources usage improvement results

Profiling tool: Scalene

To find the bottlenecks of our ICT performance, we need to profile our application in the code-line level. Therefore, we are able to monitor and analyze the execution time and memory consumption of each line of the code, other than the whole modules and functions. We also need to perform the profiling separately for the native and developed code. To do this, we adopted Scalene profiling tool, that provides our aforementioned requirements. Any other profiling tool that is capable of providing these required information can be used for this end.

Scalene is a high-performance CPU, GPU and memory profiler for Python [3]. This tool performs profiling at the code-line level and as opposed to many of the other profiling tools, it supports use of the *multiprocessing* library. To profile the memory usage, Scalene indicates the specific lines of code responsible for memory growth, using an included specialized memory allocator. It separates out the percentage of memory consumed by native code from the usage of developed code. This report helps to either focus on optimizing our developed code or detect a performance problem in a used library.

[3] `https://github.com/plasma-umass/scalene`

Scalene also reports a metric named, "copying volume" (MB/sec). This metric illustrates the speed of copying, which is quite an expensive action in terms of memory usage. The high volume of copying or a fluctuated trend of copying volume over time often indicates an issue or some performance problems related to memory usage [66]. The issues may arise from the inefficient implementation of the code or the open source libraries that are used. Further, Scalene provides CPU profiling as well. It reports CPU time spent on native/developed codes. It also separates out system time to recognize input/output bottlenecks. If there is a GPU, Scalene reports GPU time as well.

Results of execution time reduction

We use the methodology described in Section 3.3 to optimize the ICT system in terms of time. There is one parallelizable module (*Representation*) in the pipeline that is accountable for more than half of the execution time (66%). Table Table 4.5 illustrates the percentage of execution time allocated to each module of the pipeline. We calculate these execution time based on running the pipeline on the scenario no.2 of the validation set of AIC20 benchmark (see Section 4.1).

A bottleneck is discovered in the 'Representation' module when we execute our ICT. In this module there is a loop over all the videos and the processing is being performed independently for each camera. A way to reduce the execution time of this module, is a parallelization approach to execute the iterations in parallel, consequently, use the computation power efficiently. Thus, a possible solution to this end could be multi-threading. However, using multi-threads in our application that is implemented in Python would not bring in better performance due to the Global Interpreter Lock (GIL) [67]. Therefore, we use the *multiprocessing* package [4] in Python, that spreads the processes in parallel using multiple cores [68]. We create a child process per camera. Each child process executes the same task on the frames associated with a camera. The processes don't wait for each other to be complete, instead they use different processors for doing their tasks.

[4]https://docs.python.org/3/library/multiprocessing.html

Table Table 4.5: Execution time of each module of the pipeline. The experiment is run on scenario no.2 of the validation set of AIC20 benchmark on a server with a RTX 6000 GPU. The execution time of the most time consuming module is shown in bold

	Pre-processing	Representation	Re-identification	Trajectory creation
Execution time (s)	119.4	**435.1**	33.1	71.3
% of time	18.2	**66**	5	10.8

Table Table 4.6: The impact of parallelizing the Representation module on the execution time and the speed of the module and the whole pipeline. The percentage of the increased speed of the pipeline is shown in bold

	ICT	ICT + parallelization	% of improvement
Exe time of 'Representation'(s)	435.1	176	247
Total time(s)	659	400	180
FPS	12.2	22.5	**180**

We decrease the execution time of this module from 435 sec to 176 sec, using multi processing, which introduces around 2.5 times of speedup. As we can see the results in Table Table 4.6, the total execution time dropped to 399 sec. So we increased the end-to-end processing speed 1.8 times, from 12 FPS to 22 FPS. In this table, the first row compares the execution time of the 'Representation' module, with and without parallelization, and the percentage of the improvement. The second row delivers the same information of the whole ICT pipeline. The last row presents the processing speed (FPS) of the whole pipeline, before and after parallelization.

Results of memory consumption reduction

We use the methodology described in Section 3.3.2 to achieve memory efficiency. We rely on Scalene to validate our methodology and find the massive memory consumption sources. Note that due to memory constraints, we run the code on only the scenario no.2 of the validation set, described in Section 4.1.

The results of Scalene include the CPU, GPU and memory usage associated to each line of the code if there are considerable usages. An example of Scalene results is shown in Figure Figure 4.7. In this figure, the first three columns indicate the percentage of CPU time

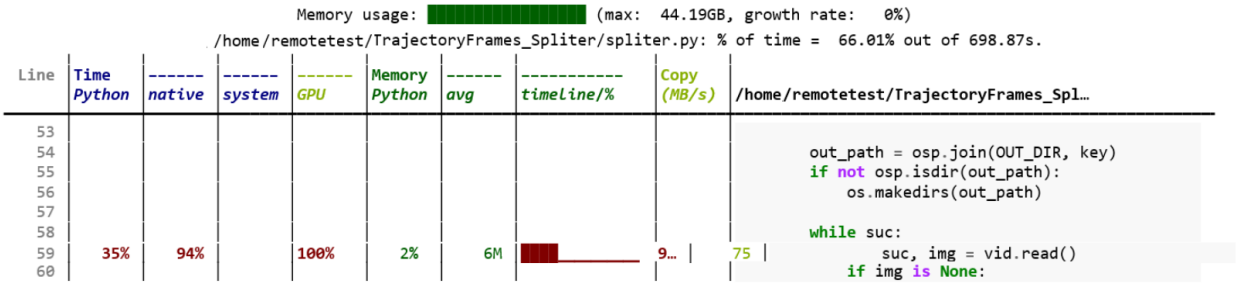


Figure Figure 4.7 A part of profiling results of the implemented ICT, using Scalene (OpenCV is used for video capturing).

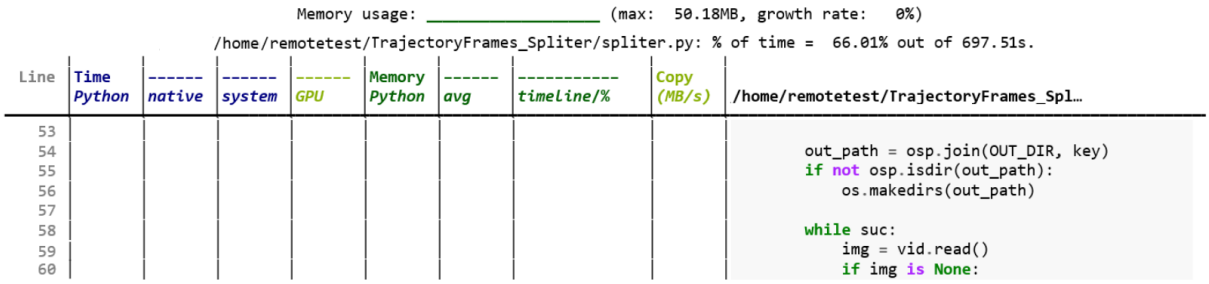


Figure Figure 4.8 The profiling result of the same part of the implemented ICT represented in Figure Figure 4.7, after replacing OpenCV with VidGear.

spent on developed code, native code and I/O actions, respectively. The next column shows GPU time spent on running each line of the code. Following, we have "Memory Python" and "avg", that impart how much of memory consumption was from developed/native code, respectively. The next column, "timeline", represents the memory usage over time. The last column expresses the copying volume per second. Scalene also provides general information on top of the tables, including the maximum memory usage and the percentage of the total time consumed by this section of the code.

Since Scalene performs profiling per function and line, by running it on our code we could identify that the most of the memory is consumed by a function, where it simply reads the frames of a video captured by an external library (See Figure Figure 4.7). For instance, in the 59th line the copy volume indicates that a considerable number of copying is happening (75 MB/sec). As there is memory consumption in native area (i.e., 'avg' column), we can conclude that the issue comes from the OpenCV library [5] we use for video capturing. To avoid this event we replace the OpenCV with a more efficient library named VidGear [6]. VidGear is built on top of many state-of-the-art specialized libraries like OpenCV, FFmpeg, ZeroMQ, picamera, starlette, yt_dlp, pyscreenshot, aiortc and python-mss at its backend[7]. It enhances the existing capabilities and performance of these libraries. Figure Figure 4.8 represents the profiling result, after using VidGear. The copy volume in line 59 has become zero, showing that we entirely removed the redundant copying. We obtain this improvement while we get the same performance in terms of IDF1, IDP and IDR and we have no noticeable change in the execution time. We drastically drop the maximum memory usage by a three orders of magnitute, from 44.19 GB to 50.18 MB. The results are shown in Table Table 4.7. In this table, the first and second rows show the effect of using different libraries on the speed

[5]https://pypi.org/project/opencv-python/
[6]https://abhitronix.github.io/vidgear/latest/
[7]https://abhitronix.github.io/vidgear/v0.2.4-stable/

Table Table 4.7: The impact of replacing OpenCV with VidGear on the power consumption of the ICT.

	ICT(OpenCV)	ICT(VidGear)	Improvement
Inadvertent Copy(MB/s)	75	0	100%
Max memory usage(MB)	**44190**	**50.2**	**0.001**×
Exe time(s)	698.7	697.5	0

of inadvertent copying, and the maximum memory usage of the ICT, respectively. The last row presents the effect on the ICT's execution time.

This way, we could detect a problem arising from the OpenCV library and achieve a huge memory saving by alternating this library. Even though OpenCV is a quite popular library in video processing projects, we revealed that it is not efficient in terms of memory usage. There is no surprise to claim that this substantial reduction in memory usage would solve one of the main restrictions of running ICTs on the edge-devices with limited resources.

4.3.3 Results on Cysca's videos

We used videos recorded in Cysca's parking lot to evaluate our method in a new environment. We need to divide the camera views into zones and create camera link models based on the possible paths to test our ICT in these new scenes. Figure Figure 4.9 illustrates sample frames of the results; in the figure when the vehicle enters the parking lot it gets the global ID in the west camera, and when it appears in the north camera it receives the same ID. The vehicle keeps the same ID in both cameras during the time it is parked, and the ID remains unchanged until the vehicle starts moving again and exits the parking lot. Note that other parked vehicles do not share global IDS in two cameras, because the ID matching works based on the vehicle trajectories, so they don't get their global IDs, unless they start moving. As these vehicles did not move since the video started, they are not assigned to their global IDs yet.

Figure Figure 4.9 Sample results on Cysca's test videos. The three scenes are captured in about 450 frame span. In the first scene. There is vehicles#0 that enters the parking lot. In the second scene the vehicle is parked having the same global IDs in both cameras. In the third and fourth scene we can see this vehicle that starts moving again and exits the parking lot while it keeps the same global ID 0.

4.3.4 Results on videos under difficult weather conditions

We evaluated our method in difficult weather conditions using the synthetic dataset we created. In the videos with adverse weather, the rain streaks and snowflakes appears on the vehicles resulting in variations in the texture and color of the vehicle. They may also cover some deterministic parts of the vehicles such as lights, vehicle model, etc. that are the detailed characteristics features used in Re-ID models. In addition, the effect of lighting on the white snowflakes result in extensive light reflection on the vehicle windows and bodies, causing a large part of the vehicle to disappear for a couple of subsequent frames, which can fool the Re-ID model. These appearance variations make the re-identification task more challenging.

Table Table 4.8 shows the result of our ICT evaluation on these videos. In this table, the first row presents the results on the standard AIC20 dataset, with normal weather condition, and the second row shows the results on our synthetic dataset with challenging weather. Regarding the evaluation result the effectiveness of our ICT is reduced by about 3% in terms of IDF1 metric, comparing to the normal condition.

Table Table 4.8: Our proposed ICT evaluation results on the videos with difficult weather conditions compared to normal conditions.

Dataset	Noise	IDF1(%)	IDP(%)	IDR(%)
AIC20, Scenario no.2	None	82.9	75.1	93
AIC20, Scenario no.2	Synthetic snow and rain	79.8	72.3	89

4.4 Summary

In summary, in this chapter we

- reviewed the dataset and benchmark we used for our experiments and evaluations, including AIC20 dataset, videos from a parking lot provided by our industrial partner, Cysca Technologies, and our proposed synthetic dataset with adverse weather conditions.
- described the metrics used to measure the performance of our ICT, as well as the testing environments configuration.
- validated our proposed ICT quantitatively and visually using the introduced benchmarks, and presented the impact of using the new Re-ID model and the hybrid distance metric on the ICT effectiveness. Compared to the base pipeline, we achieved about 2 percent improvement, reaching an IDF1 score of 72.1% on AIC20 dataset.
- applied our proposed strategy of resource usage reduction on our ICT, resulting in a three orders of magnitude reduction in memory consumption, and 1.8× increment in processing speed.
- evaluated our ICT in adverse weather conditions using our proposed synthetic dataset. Our ICT showed resilience only to some extent, and its IDF1 score reduced by 3%.

CHAPTER 5 CONCLUSION

This chapter concludes the study by summarizing our work in relation to the research objectives and reviewing our contribution. We also discuss the limitations of the study and proposes recommendations for future research.

5.1 Summary of Works

This study aimed to present an efficient inter-camera vehicle tracking system that is applicable in urban settings with various road and path structures, and perform in real-time speed. We relied on a state-of-the-art MCT method to implement our ICT pipeline. Then, we utilized a robust trajectory distance definition for trajectory association to enhance the tracker effectiveness. The distance is defined based on k-reciprocal nearest neighbors, to better handle the inter-class similarity and intra-class variability. We also replaced the ReID model used in the base pipeline, with an ensemble ReID model that provides more discriminating appearance features. Our results showed that these modifications improved the ICT effectiveness noticeably. Since the pipeline's implementaion is modular, it can easily adapt to other strategies for considering spatial-temporal information and constraints, and also benefit from the future improved ReID models or camera link models.

Furthermore, we proposed a methodology to improve the resource usage of the ICTs. Our experimental results showed that our system throughput and memory footprint is significantly improved using our proposed resource usage reduction strategy. This significant reduction in resource usage is a one big step forward in running this ICT on memory-limited and edge devices.

Our method can be used in both roads and parking lots. Besides, the camera link models used in our method are built based on the trajectories patterns of the vehicles, therefore it is easily adaptable to the new environment with different architecture of camera networks.

We validated our ICT on an annotated dataset that is commonly used for ICT systems evaluations, i.e., AIC20, and on videos from a parking lot. In addition, we created a synthetic annotated dataset with snowy and rainy weather to evaluate our method under difficult weather conditions. The results showed that our ICT preserved its effectiveness to a certain extent, under challenging conditions.

5.2 Limitations

The more accurate SCT used for tracking vehicles in each camera, the more efficient ICT is obtained, because the SCT results are used as input in our ICT. Therefore, the performance of our ICT is strongly dependent on the SCT performance.

Although our ICT achieves effectiveness that is comparable to state-of-the-art in terms of IDF1 score, it cannot preserve its effectiveness under difficult weather conditions. However, our proposed synthetic snowy dataset brings in the opportunity to try different solutions to preserve the performance under challenging conditions, and evaluate the results.

There is also room for ICT effectiveness improvement, specifically, when there are vehicles with very similar appearance moving extremely closely and being occluded by each other or other vehicles repetitively.

5.3 Future Research

As a solution for the inter-class similarity problem, we can consider using the delicate local features of the vehicle images along with their global features for re-identification. There are discriminating details on specific regions of the vehicles, such as window stickers, and rims, making the vehicles with the same model and color distinguishable. Being aware of the orientation of the vehicle image, the desired regions of the vehicles can be easily found. Khorramshahi et al. [69] proposed an orientation estimation network that can be used to this end.

As a work around for maintaining the ICT performance in challenging weather conditions, we may explore snow and rain removal methods that remove the effect of snow and rain from the vehicle images, while preserving real-time speed of processing.

Furthermore, the system throughput can still be improved using model compression techniques that reduce the size of our ReID models, while maintaining their effectiveness. Using compression techniques make the models more fit for memory-limited devices.

Another perspective to improve the ICT results is providing it with more subtle input. For instance, one can combine the SCT methods with our ICT, that effectively handle occlusions. The less trajectory fragmentation and mismatch in SCT results, the less false matches appears in our ICT.

REFERENCES

[1] B. Dickson, "What is computer vision?" [Online]. Available at https://bdtechtalks.com/2019/01/14/what-is-computer-vision/.

[2] A. J. O'Toole, P. J. Phillips, F. Jiang, J. Ayyad, N. Penard, and H. Abdi, "Face recognition algorithms surpass humans matching faces over changes in illumination," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 9, pp. 1642–1646, 2007.

[3] T. Huang, *Computer vision: Evolution and promise*. Meyrin, Switzerland: Cern, 1996.

[4] A. Rizzoli, "Most popular computer vision applications and use cases in 2022." [Online]. Available at https://www.v7labs.com/blog/computer-vision-applications.

[5] H.-M. Hsu, T.-W. Huang, G. Wang, J. Cai, Z. Lei, and J.-N. Hwang, "Multi-camera tracking of vehicles based on deep features re-id and trajectory-based camera link models." in *CVPR Workshops*, 2019, pp. 416–424.

[6] C. Liu, Y. Zhang, H. Luo, J. Tang, W. Chen, X. Xu, F. Wang, H. Li, and Y.-D. Shen, "City-scale multi-camera vehicle tracking guided by crossroad zones," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4129–4137.

[7] J. Ye, X. Yang, S. Kang, Y. He, W. Zhang, L. Huang, M. Jiang, W. Zhang, Y. Shi, M. Xia *et al.*, "A robust mtmc tracking system for ai-city challenge 2021," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4044–4053.

[8] H.-M. Hsu, J. Cai, Y. Wang, J.-N. Hwang, and K.-J. Kim, "Multi-target multi-camera tracking of vehicles using metadata-aided re-id and trajectory-based camera link model," *IEEE Transactions on Image Processing*, vol. 30, pp. 5198–5210, 2021.

[9] A. K. Haghighat, V. Ravichandra-Mouli, P. Chakraborty, Y. Esfandiari, S. Arabi, and A. Sharma, "Applications of deep learning in intelligent transportation systems," *Journal of Big Data Analytics in Transportation*, vol. 2, no. 2, pp. 115–145, 2020.

[10] J. Cai, J. Deng, M. U. Aftab, M. S. Khokhar, R. Kumar *et al.*, "Efficient and deep vehicle re-identification using multi-level feature extraction," *Applied Sciences*, vol. 9, no. 7, p. 1291, 2019.

[11] X. Liu, W. Liu, T. Mei, and H. Ma, "Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 645–658, 2017.

[12] Y. Bai, Y. Lou, F. Gao, S. Wang, Y. Wu, and L.-Y. Duan, "Group-sensitive triplet embedding for vehicle reidentification," *IEEE Transactions on Multimedia*, vol. 20, no. 9, pp. 2385–2399, 2018.

[13] N. Jiang, Y. Xu, Z. Zhou, and W. Wu, "Multi-attribute driven vehicle re-identification with spatial-temporal re-ranking," in *2018 25th IEEE international conference on image processing (ICIP)*. IEEE, 2018, pp. 858–862.

[14] J. Zhu, H. Zeng, Z. Lei, S. Liao, L. Zheng, and C. Cai, "A shortly and densely connected convolutional neural network for vehicle re-identification," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 3285–3290.

[15] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8797–8806.

[16] W. Chen, L. Cao, X. Chen, and K. Huang, "A novel solution for multi-camera object tracking," in *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 2329–2333.

[17] L. Patino and J. Ferryman, "Multicamera trajectory analysis for semantic behaviour characterisation," in *2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2014, pp. 369–374.

[18] M. Evans, C. J. Osborne, and J. Ferryman, "Multicamera object detection and tracking with object size estimation," in *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*. IEEE, 2013, pp. 177–182.

[19] L. Patino and J. Ferryman, "Multicamera trajectory analysis for semantic behaviour characterisation," in *2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2014, pp. 369–374.

[20] Y. Wang, "Distributed multi-object tracking with multi-camera systems composed of overlapping and non-overlapping cameras," Ph.D. dissertation, The University of Nebraska-Lincoln, 2013.

[21] X. Li, Z. Zhou, P. L. Mazzeo, S. Ramakrishnan, and P. Spagnolo, "Object re-identification based on deep learning," in *Visual Object Tracking with Deep Neural Networks*. intechopen, 2019.

[22] S. D. Khan and H. Ullah, "A survey of advances in vision-based vehicle re-identification," *Computer Vision and Image Understanding*, vol. 182, pp. 50–63, 2019.

[23] D. Cornelisse, "An intuitive guide to convolutional neural networks." [Online]. Available at https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/.

[24] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1318–1327.

[25] S.-H. Lee, M. Bang, K.-H. Jung, and K. Yi, "An efficient selection of hog feature for svm classification of vehicle," in *2015 International Symposium on Consumer Electronics (ISCE)*. IEEE, 2015, pp. 1–2.

[26] D. Zapletal and A. Herout, "Vehicle re-identification for automatic video traffic surveillance," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 25–31.

[27] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[28] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[29] H. Wang, J. Hou, and N. Chen, "A survey of vehicle re-identification based on deep learning," *IEEE Access*, vol. 7, pp. 172 443–172 469, 2019.

[30] M. Nafzi, M. Brauckmann, and T. Glasmachers, "Methods of the vehicle re-identification," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2020, pp. 516–527.

[31] Y. Sun, L. Zheng, Y. Yang, Q. Tian, and S. Wang, "Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline)," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 480–496.

[32] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei, "Circle loss: A unified perspective of pair similarity optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6398–6407.

[33] Y. Zhang, Q. Qian, C. Liu, W. Chen, F. Wang, H. Li, and R. Jin, "Graph convolution for re-ranking in person re-identification," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 2704–2708.

[34] S. He, H. Luo, P. Wang, F. Wang, H. Li, and W. Jiang, "Transreid: Transformer-based object re-identification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 013–15 022.

[35] X. Li, Z. Zhou, P. L. Mazzeo, S. Ramakrishnan, and P. Spagnolo, "Object re-identification based on deep learning," in *Visual Object Tracking with Deep Neural Networks*. intechopen, 2019.

[36] Z. Zhou, X. Li, and M. Qiu, "A car face parts detection algorithm based on faster r-cnn," in *CICTP 2018: Intelligence, Connectivity, and Mobility*. American Society of Civil Engineers Reston, VA, 2018, pp. 295–304.

[37] X. Liu, W. Liu, T. Mei, and H. Ma, "Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 645–658, 2017.

[38] N. Jiang, Y. Xu, Z. Zhou, and W. Wu, "Multi-attribute driven vehicle re-identification with spatial-temporal re-ranking," in *2018 25th IEEE international conference on image processing (ICIP)*. IEEE, 2018, pp. 858–862.

[39] Z. Wang, L. Tang, X. Liu, Z. Yao, S. Yi, J. Shao, J. Yan, S. Wang, H. Li, and X. Wang, "Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 379–387.

[40] Y.-G. Lee, J.-N. Hwang, and Z. Fang, "Combined estimation of camera link models for human tracking across nonoverlapping cameras," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 2254–2258.

[41] Z. Tang, G. Wang, H. Xiao, A. Zheng, and J.-N. Hwang, "Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic

features," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 108–115.

[42] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8797–8806.

[43] H.-M. Hsu, Y. Wang, and J.-N. Hwang, "Traffic-aware multi-camera tracking of vehicles based on reid and camera link model," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 964–972.

[44] M. Wu, Y. Qian, C. Wang, and M. Yang, "A multi-camera vehicle tracking system based on city-scale vehicle re-id and spatial-temporal information," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4077–4086.

[45] M. Nikodem *et al,*, "Multi-camera vehicle tracking using edge computing and low-power communication," *Sensors*, vol. 20, no. 11, p. 3334, 2020.

[46] G. Wang, Y. Wang, H. Zhang, R. Gu, and J.-N. Hwang, "Exploit the connectivity: Multi-object tracking with trackletnet," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 482–490.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[49] J. Gao and R. Nevatia, "Revisiting temporal modeling for video-based person reid," *arXiv preprint arXiv:1805.02104*, 2018.

[50] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[51] M. Everingham, S. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[52] G. Hua and H. Jégou, *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II*. Springer, 2016, vol. 9914.

[53] N. Lawrence and A. Hyvärinen, "Probabilistic non-linear principal component analysis with gaussian process latent variable models." *Journal of machine learning research*, vol. 6, no. 11, 2005.

[54] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[55] N. D. Lawrence, "Learning for larger datasets with the gaussian process latent variable model," in *Artificial intelligence and statistics*. PMLR, 2007, pp. 243–250.

[56] H. Luo, Y. Gu, X. Liao, S. Lai, and W. Jiang, "Bag of tricks and a strong baseline for deep person re-identification," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019, pp. 0–0.

[57] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[58] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*. Springer, 2016, pp. 499–515.

[59] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M.-C. Chang, X. Yang, L. Zheng, A. Sharma *et al.*, "The 4th ai city challenge," in *CVPR Workshop*, 2020.

[60] M. Naphade, Z. Tang, M.-C. Chang, D. C. Anastasiu, A. Sharma, R. Chellappa, S. Wang, P. Chakraborty, T. Huang, J.-N. Hwang *et al.*, "The 2019 ai city challenge." in *CVPR Workshops*, vol. 8, 2019.

[61] Y. Yao, L. Zheng, X. Yang, M. Naphade, and T. Gedeon, "Simulating content consistent vehicle datasets with attribute descent," in *European Conference on Computer Vision*. Springer, 2020, pp. 775–791.

[62] Z. Tang, M. Naphade, S. Birchfield, J. Tremblay, W. Hodge, R. Kumar, S. Wang, and X. Yang, "Pamtri: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 211–220.

[63] M. Naphade, M.-C. Chang, A. Sharma, D. C. Anastasiu, V. Jagarlamudi, P. Chakraborty, T. Huang, S. Wang, M.-Y. Liu, R. Chellappa *et al.*, "The 2018 nvidia ai city challenge," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 53–60.

[64] N. Yaghoobi Ershadi, "Improving vehicle tracking rate and speed estimation in dusty and snowy weather conditions with a vibrating camera," *PLoS one*, vol. 12, no. 12, p. e0189145, 2017.

[65] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European conference on computer vision*. Springer, 2016, pp. 17–35.

[66] E. Berger, "Scalene: a high-performance, high-precision cpu+gpu+memory profiler for python." [PyCon US, 2021]. Available at https://pyvideo.org/events/pycon-us-2021.html.

[67] T. Khot, "Parallelization in python," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 23, no. 3, pp. 56–58, 2017.

[68] T. Ziadé *et al.*, *Expert Python Programming*. Packt Publishing, 2008.

[69] P. Khorramshahi, A. Kumar, N. Peri, S. S. Rambhatla, J.-C. Chen, and R. Chellappa, "A dual-path model with adaptive attention for vehicle re-identification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6132–6141.