UNIVERSITÉ DE TOURS

École Doctorale MIPTIS

Laboratoire d'Informatique Fondamentale et Appliquée de Tours (LIFAT)

EA 6300 - ERL CNRS 7002

THÈSE présenté par :

Quoc-Cuong LE

soutenue le : 07 Septembre 2020

pour obtenir le grade de : Docteur de l'université de Tours

Discipline/ Spécialité : Informatique

OBJECT DETECTION AND TRACKING IN COLLABORATIVE AND DISTRIBUTED CAMERA NETWORK

Remerciements

Je tiens à remercier Monsieur Donatello CONTE, Professeur à l'Université de Tours, qui m'a encadré tout au long de cette thèse et qui m'a fait partager ses brillantes intuitions. Qu'il soit aussi remercié pour sa gentillesse, sa disponibilité permanente et pour les nombreux encouragements qu'il m'a prodiguée.

Je remercie Monsieur Moncef HIDANE, Professeur à l'INSA Centre Val de Loire, qui m'a co-encadré avec Professeur CONTE, pendant ma thèse et qui m'a donné les conseils, ainsi que les retours critiques mais pertinants au long des travaux que j'ai réalisé pendant mes années de thèse, et qui m'a initié au carrière de recherche, et particulièrement au traitement d'image, son cours j'ai découvert ma passion en vision par Ordinateur. J'apprécie tout ses conseils pendant ma dernière année à l'INSA. Qu'elle en soit remerciée.

J'adresse tous mes remerciements à Monsieur Thierry CHATEAU, Professeur à l'Université de Clemont Auvergne, ainsi qu'à Madame Séverine DUBUISSON, Professeur à l'Institut d'Aix-Marseille, de l'honneur qu'ils m'ont fait en acceptant d'être rapporteurs de cette thèse.

J'exprime ma gratitude à Monsieur Sébastien ADAM, Professeurs à l'Université de Rouen, et à Monsieur RAMEL Jean-Yves, qui ont bien voulu être examinateurs.

Je souhaite remercier Professeur TRAN THANH VAN et sa femme, Professeuse LE KIM NGOC. Les deux grands rechercheurs m'ont aidé à réaliser mes aventures en France, qui m'ont encouragé de poursuire ma passion en recherche, qui m'ont partagé leur sagesse.

Je remercie Van Ut TRAN, et sa famille, Dinh Cong NGUYEN, Luong Phat NGUYEN, Kieu Diem HO, Xuan Dung LE et mes collègues du LIFAT qui j'ai partagé tous ces moments de doute et de plaisir.

Un grand merci aussi à tous les membres de l'association Touraine-Vietnam et en particulier à Jocelyn ROBIN, Jean-Jaques ROUSSELLE, Aude-Hélène GEORGET, Laura PASQUET, qui m'ont accueilli et organisé les activités sociatifs culltures.

J'adresse mes remerciements à Krytal REARICK, Daniels SPINKS, Taylor WOODFILL, Valérie MISTRETTA, Abigail Klinedinst, Guy STRUDWICK, Christopher MURPHY, qui m'ont supporté pendant les momments difficiles, m'a appris votre langue et m'a partagé des joyeux et inoubliables moments. Je vous appricie.

Finalement, je remercie ma famille au Vietnam, mes parents, ma soeur et mon frère, qui toujours m'a soutenu à n'importe quel moments au long de mes années de thèse.

Pour finir cette page, j'écris un citation de l'un des plus grands inventeurs qui m'a guidé dans mes études en France.

"Our greatest weakness lies in giving up. The most certain way to succeed is always to try just one more time." Thomas A. Edison

Résumé

Les travaux présentés dans cette thèse abordent les problématiques de détection et suivi d'objets, en utilisant un système de caméras collaboratives. L'idée principale de l'utilisation de plusieurs caméras pour réaliser le suivi est de résoudre les problèmes d'occultation que les méthodes de suivi de mono-caméra sont incapables de régler. Pour s'adapter des critères dans plusieurs applications de surveillance, nos travaux se concentrent sur le problème de suivi en ligne de plusieurs objets dans le contexte de plusieurs caméras synchronisées et dont les champs de vue sont chevauchent. Dans le cas de notre étude, les axes suivants ont été étudiés : premièrement, utiliser plusieurs caméras pour suivre une seule cible; deuxièmement, suivre plusieurs d'objets simultanément; finalement, réidentifier les objets qui réapparaissent ultérieurement dans le champs de vue. Dans les conditions où les tâches de suivi se font sur une scène en plein air, l'apparence des objets (forme, couleur, texture, ...) change. Les changements sont dus aux conditions de luminosité variant à l'extérieur, des mouvements des objets eux-mêmes. Souvent, les performances de suivi sont dégradées à cause de la perte de leurs cibles. Nous avons développé des algorithmes de suivi avec multi-caméras qui permettent à chaque caméra de participer au processus de suivi des autres caméras dans le réseau.

En détail, notre première contribution est une plateforme de suivi avec plusieurs caméras basée sur les filtres particulaires qui permet à une caméra de collaborer avec les autres caméras dans le cas où la cible est occultée par l'environnement ou l'autres cibles. Le modèle d'apparence d'objet est une représentation parcimonieuse où les différentes apparences d'une cible sont représentées by une combination de plusieurs patches d'image de référence contenue dans un dictionnaire. Notre deuxième contribution concerne des problèmes d'occultation mutuelle entre objets. En pratique, les objets souvent s'occultent mutuellement, particulièrement dans les scènes de foules. En adaptant une approche de suivi de multi-objet avec le Processus de Décision Markovien au contexte d'un système de caméras collaboratives, notre algorithme de suivi avec multi-caméras permet de résoudre des problèmes d'occultation dans le suivi par une seule caméra. Notre troisième contribution concerne de la réidentification des cibles d'une vue à l'autre. Nous avons reformulé le problème d'affectation de cibles entre deux vues comme une problème du plan de transport optimal non-équilibré. Nous avons ensuite étendu les résultats d'affectation de cibles dans les paires de caméras à notre algorithme de suivi par multi-caméra. En outre, nous étudions les caractéristiques d'apparence qui permets de réidentifier des objets dans différentes vues comme l'histogramme de couleurs, des points d'intérêt Lucas-Kanade ou des caractéristiques extraites par des réseaux convolutifs profonds.

En ce concerne des expérimentations, nous avons évalué notre algorithme par les métriques communnes sur les bases de données publiques. Les résultats expérimentaux ont montré la pertinence de nos algorithmes de suivi multi-caméras par rapport à une seule caméra, ainsi que l'impact des différentes caractéristiques sur la performance de suivi de notre approche de suivi par multi-caméras.

Mots clés : Detection d'objets, Suivi de mono-objet, Suivi de multi-objet, Suivi de multi-target multi-camera, Apprentissage profond de caractéristiques, Transport optimal.

Abstract

The work presented in this thesis concerns the problem of visual multiple object tracking using a system of collaborative cameras. The main idea of using a multi-camera system in tracking is to solve occlusion problems, which single-camera tracking methods are unable to solve. With multiple automated surveillance applications in mind, our work focuses on the problem of online multi-object tracking in a multi-camera system in which fields of view are overlapped, and video frames are synchronized. In the case of our study, the thesis includes the following objectives: firstly, to track a single object in a multi-camera system; secondly, to track multiple objects simultaneously; finally, to re-identify objects which possibly reenter the fields of view of the cameras multiple times. In outdoor tracking scenes, objects often change their appearances, including their shape, their size, and their texture. The changes are due to the varying lighting condition and the movement of the objects themselves. This causes tracking algorithms to lose their targets frequently, and therefore degrades tracking performance. We developed multi-camera tracking algorithms that allow each camera to participate in the overall tracking process of the network to improve its tracking result.

In detail, our first contribution is a multi-camera tracking framework based on particle filters that allows a camera to collaborate with other cameras when targets are occluded by the environment or other targets. The model used for object appearance, in our framework, is the sparse representation, in which the variant appearances of a target are represented by a combination of reference image patches contained in a dictionary. Our second contribution addresses the problem of mutual occlusion between objects. In practice, objects are often occluded, especially in crowded scenes. By adopting a Markov Decision Process multi-object tracking algorithm to the context of multiple collaborative cameras, our tracking algorithm mainly solves the problems of occlusion occurring in single-camera tracking. Our third contribution concerns re-identifying targets across cameras. We reformulated the assignment problem of targets between two views as an unbalanced optimal transport problem. The target assignments in pairs of cameras are then adapted to our multi-camera tracking algorithm. Additionally, we studied multiple appearance features that allow the multi-camera system to re-identify multiple targets in different views such as color histogram, Lucas-Kanade keypoints, or deep characteristic features extracted by convolutional neural networks.

Concerning experimentation, we evaluated our algorithms by multiple common metrics on public multi-camera video databases. Our experiments showed the relevance of our multi-camera tracking algorithms to single-camera ones, as well as the impact of different characteristic features on the tracking performance of our multi-camera tracking approach.

Keywords : Object Detection, Single-Object Tracking, Multi-Object Tracking, Multi-Target Multi-Camera Tracking, Deep Feature Learning, Optimal Transport.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Since the last decades, technological advances in computers and cameras have brought image technologies into many aspects of real life. With the easy accessibility to cameras and massive data storage systems, single or multiple camera systems are built for many, either civil or military applications. One of the most popular applications of camera systems is security surveillance. Since the availability of camera systems, security surveillance with camera systems has been widely deployed to prevent criminal activities in public or private areas. Additionally, as the terror acts of extremist or terrorist groups have happened around the world, especially for the last five years, and big cities such as New York, London, Paris, Barcelona seemed to be the vulnerable targets. The public safety concern has been risen at the highest level than ever. Consequently, the security camera surveillance system has become so demanding and essential for police authorities to ensure the safety of their citizens.

In practice, video surveillance systems have demonstrated their effectiveness to help police force quickly identify and track down suspects or criminals in investigations. In 2013, when the Boston bombing happened, the suspects have been identified quickly because of the video surveillance system in New York City. However, it took the law enforcement force in Boston three years to finally end this dramatic manhunt. In China, the authorities deployed the SkyNet Project, a national surveillance system, which consists of more than 20 million cameras set up in public spaces and a massive computation center to process millions of images a day gathered from those cameras. This surveillance system is expected to detect in prior crimes that potentially happen in public and to observe the behavior of their citizens, which is part of the social credit program in this country. A couple of years ago, Amazon launched new stores in the US, which is named Amazon Go. The specialty of this new kind of store is that shoppers are not required to check out when they leave. Therefore, people do not have to wait in line; all that they need is to install Amazon Go apps linking with a credit card, pick up their items, and walk out. A complex sensor system in the store can detect when products are taken and returned to the shelves, meanwhile, a camera network monitors their customers and add the products they picked to their virtual cart on their phone apps. As we can see, the success of this initiative firmly relies on its automatic surveillance camera system. Amazon Go stores have already opened in several cities in the US, and Amazon wants to expand its future of retail in the UK soon. Beyond land security, camera surveillance systems are used for marine safety, in which they are usually installed on vessels or coast guard boats to observe any suspicious movement and detect pirates or drug smuggling ships.

Besides the apparent advantages of security camera systems, there are several setbacks such as most of the camera surveillance systems are centralized. In other words, this kind of system comprises a network of cameras set up around outdoor or indoor spaces connecting to a surveillance center where all video streams are gathered, recorded in a database. In most cases, those videos are streamed directly on plenty of screens, and one or several security agents watch all these screens to oversee the surveillance areas regularly. The general architecture of camera surveillance systems is shown in the Figure 1.1.

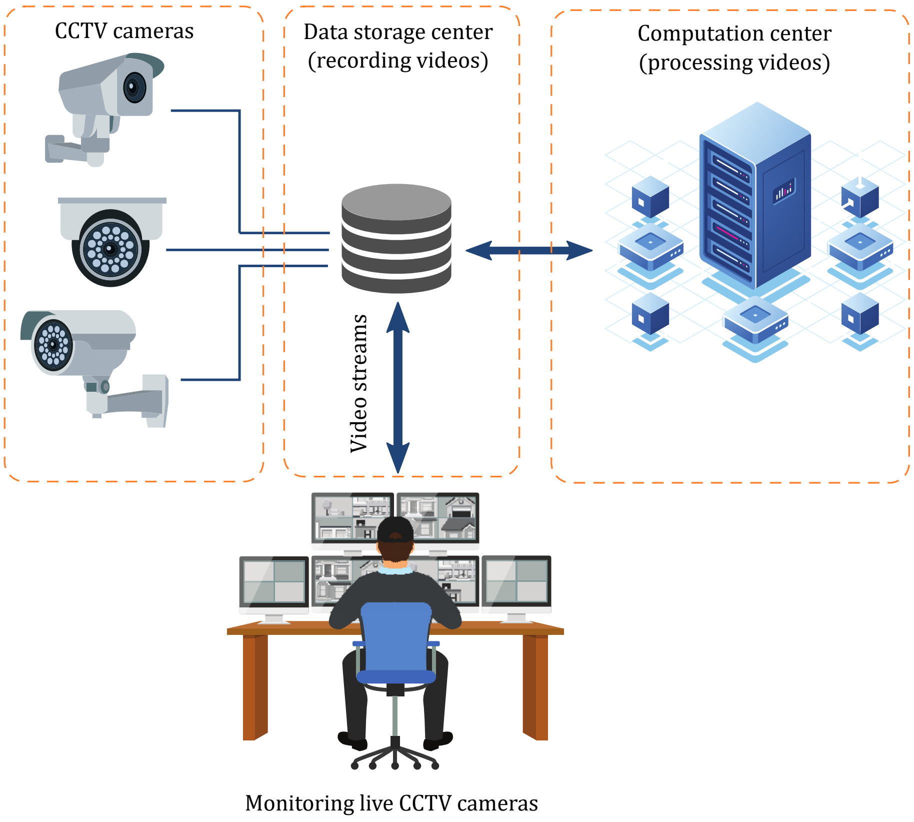


Figure 1.1 – The general architecture of camera surveillance systems

However, a study in 2002 conducted by Brandon C. Welsh and David P. Farrington [222] showed the ineffectiveness of CCTV[1] on reducing crimes. That is explained by the fact that not all of those cameras are truly monitored. In reality, it is hard for one security agent to watch over lots of screens in a long time. It is evident that by the end of the day, they can be distracted and eventually miss events, which potentially leads to a crime. On the other hand, in the case of reviewing all video records for crime investigation, fast-

[1]Closed-Circuit Television

forwarding every single video is an exhausting and time-consuming work. In some cases, to lower the cost of security, only several available screens are used to display all video streams. The images from cameras are switched periodically and not always monitored constantly. Another study on how security CCTV cameras impact crimes in the urban environment led research on about 1000 camera networks [189] in Chicago city in 2013. The effectiveness of surveillance cameras relies on whether they are being overseen.

As a result of the increasing demand for CCTV cameras for public safety, it is required an *automatic* surveillance systems without adding more human resources. This system involved three essential tasks, including detection, tracking, and re-identification, which are the aspects of the study of this thesis. Furthermore, there are many setbacks of current surveillance systems that motivate this study. One of them is the privacy concern about storing individuals' images without permission. Although we think that people who have done nothing wrong should not be worried, most people who have been asked if video surveillance systems are intruding their privacy, their answer is almost yes. Depending on countries, the legislation on privacy protection might be different. In China, the privacy concern does not seems to be a big issue, even though the international community has strongly criticized its national surveillance system as such a human right violation. Meanwhile, in Europe, the EU data protection rules, as known as the EU General Data Protection Regulation (or GDPR) [51], was adopted by the European Parliament in 2016 to increase the transparency of businesses in data collection and to protect the privacy of EU citizens. In general, the new EU data protection rules aim to give people the right to know if their personal data is collected and also the right to suspend any data collection to them or request their personal data. This new EU regulation makes a significant impact on video surveillance systems. Because most of these systems always collect images from cameras and save them into a data storage center before that information being processed for further tasks such as tracking, recognition, and re-identification. Hence, this law adds more requirements to surveillance systems, and it means that data storage should be suspended. That forces surveillance camera systems to process video streams online.

With the rapid improvement of embedded systems in computational power and low energy consumption in addition to low-cost cameras, the "intelligent cameras " or "smart cameras" are defined as the cameras which can process end-user tasks (e.g., detection, tracking, analyzing) themselves without transferring the video streams back to computation center. Unlike the conventional centralized system, a surveillance camera system built by smart cameras technically fulfills the aforementioned strict requirements on privacy. As a distributed computing system, all raw information is processed locally by cameras themselves so that this sensitive information is not being transferred or spread into networks. This means that no massive data centralized storing is required, and data are almost processed online by each camera in the network. By computing data in a distributed way, the smart camera system lowers the amount of necessary information transferring through the network. Another convenience of this system is that cameras can work independently or collaboratively within an unstable network connection such as wireless or mobile signals. Finally, with the capacity of operating on the existing IT infrastructure, this camera system is not required any additional network for the surveillance system (e.g., IP cameras), it is then easy to set up with a lower installation cost. Therefore, a distributed and collaborative camera network to detect and track objects is the main focus of this thesis.

Apart from security applications mentioned previously, videos from cameras are being served for different purposes, such as scene understanding. As one of the most crucial tasks in video processing, tracking algorithms are not only used to follow the trace of an individual for qualitative analysis but also to determine the group tracking for quantitative analysis. The tracking results, i.e., the trajectories of the objects of interest in videos are served as a relevant feature for automatic deeper analyses such as classifying videos in videos databases (e.g., YouTube, DailyMotion), indexing scenes in movies on online streaming movies services (e.g., Netflix, Amazon Prime Video, Apple TV, Hulu) or monitoring traffic at a busy street intersections.

In addition to these above applications in real world, object detection and tracking with cameras are a hot research topic for autonomous driving technologies that could apply in future applications such as robo-taxi service. In self-driving systems, besides the autonomous navigation part that drives unmanned vehicles on their road lanes and follow their given itinerary, driverless cars essentially require a complex object detection and tracking system with multiple sensors including radio-signal sensors and visual sensors (i.e., cameras) to detect immediately any obstacle on the road in any weather condition. Driving on the road does not just mean to drive from one point to another point in the city, the car has to collaborate with other vehicles which are, at the same time, participating on the traffic. All vehicles have to follow the road safety rules such as recognizing road signs, slowing down when approaching intersections or stopping at the red traffic lights. Driverless cars have to behave on the road independently and exactly as being driven by human. Human being uses eyes to observe other cars, pedestrians, road signs on the street, meanwhile driverless vehicles use cameras to do the same things. Processing video streams from cameras, the computer system on this type of vehicles is required to detect other vehicles and people in real time in order to recognize any abnormal movement on the road to avoid accidents as early as possible. As the safety of customers is the priority of any driverless car maker, object detection and tracking is an ultimate crucial task, which is also the main focus of the manufacturers in order to prevent road accidents. Being a leader in self-driving car industries, Tesla was the first automaker to release the autopilot system on its electric car (Tesla Model S) in 2014. Despite the constant improvement of the autopilot system, there have been crashes on the experiments of this new autopilot system. The first known fatal car crash occurred in Florida on a night of May 2016. According to the conclusion from the local authorities, the camera system was malfunctioning while it did not recognize the victim crossing the street. In another experiment, Tesla released a video showing its self-driving car has successfully tracked the front car and detected its abnormal slow-down and the incident ahead, then stopped the vehicle to prevent the follow-up accident. Those examples demonstrated that a detection and tracking system plays a crucial role in self-driving vehicles (Fig. 1.2).

As all the reasons mentioned above motivate this study, the main objective of this thesis is to develop novel methods of object detection and tracking via a network of collaborative and distributed cameras. The next sections detail the context of this study, the remaining challenges in the field, the prior hypotheses fixing our case study as well as our contributions.

Figure 1.2 – Detection and tracking in self-driving cars

1.2 Context of study

As part of the LUMINEUX project, this study is funded by the Région Centre, France. The LUMINEUX project focuses on the energy efficiency of urban lighting and all the following related overconsumption problems. The main goal of the LUMINEUX project is to make urban and peri-urban lighting intelligent built from an embedded vision system which can communicate between its units. This new system contributes to reducing the costs of operating and protecting the environment by lowering energy consumption. The goal of the project is to create an intelligent lighting system in cities based on the analysis of the scene resulted from the vision system. Furthermore, this intelligent vision system aims to detect and recognize abnormal, suspicious events in public place, e.g., road accidents, crime scenes, traffic violations. As a result, this project will expectedly improve road safety, as well as public safety in general.

As one of the most ongoing and trendy problems in computer vision, automatic video understanding and interpretation involve detection, tracking and recognition of objects of interest which is one of the main focuses of this study. The state-of-the-art methodologies for video understanding are developed within a powerful system using a single camera. Meanwhile, with the rapid expansion of cheap camera networks, collaborative and distributed methods for video understanding are becoming more and more interesting research topic. Hence, we aim at developing efficient collaborative and distributed solutions for network-based video understanding during this thesis. We are particularly interested in the tracking problem and targeting at new robust algorithms which improve both tracking reliability on a single camera and overall understanding of the scene captured by the camera network.

This research study entirely took place at the Laboratory of Fundamental and Applied

Computer Science of Tours (french acronym: LIFAT) (EA 6300), L'école polytechnique de Tours within the University of Tours, France.

1.3 Remaining challenges in object tracking

As mentioned previously, the goal of this study is to develop novel methods of object detection and tracking within a distributed and collaborative camera network. Since the last decade, there have been plenty of works on object detection and tracking by cameras for surveillance purposes. However, this research topic is still having challenging problems.

First of all, to perform tracking tasks, the camera system has to detect objects of interest at the first place when they enter the scene. In the literature on object detection or generic object detection, there are many problems such as: building a consistent pattern for a specific object; dealing with various shapes and textures of objects, tackling the variation of one single object in size, scale and pose, the problems from objects themselves; and other issues caused by environmental factors include camouflage (i.e., when the texture/color of the background is quite similar to objects); bad lighting condition (e.g., detecting object in the dark); or the error from recording devices (e.g., image distortion caused by camera lenses, noise generated from image sensors).

In this thesis, we are particularly interested in vehicles and pedestrians for public surveillance purposes. For street vehicles such as cars, trucks, the 3D appearance of them are consistent and unchanged, but when being recorded in 2D images, their poses, which is depending on the camera's angle and position, are directly impacting on its 2D appearance. Meanwhile, people change their shape when moving. Furthermore, individuals' physical appearance highly depends on their height, body, and clothing as well. Because of the various surveillance systems, human poses can be a difficult issue. For example, the camera surveillance systems on public transports such as buses, trains, metros usually have the top view in which the cameras mainly observe the heads of passengers.

One of the main factors impacting on the tracking performance of camera surveillance systems is the constant environmental changes. Unlike the indoor camera system, the camera surveillance systems in public places are deeply affected by the background scene changes. These changes can be the change of lighting during day and night or caused by weather (e.g., sunny or cloudy days) or the shadow of objects such as buildings, trees, advertising billboards, or the lawn color changes (for example from green to yellow), tree branches moving with winds. In outdoor cases, the color of targets is usually changed by lighting conditions, for example, an individual walking out from the shade of a tree. However, these environmental factors do not seem affecting detectors, but challenging the re-identification task later, because of the color change on targets.

Another critical problem in detection and tracking is occlusion. In surveillance videos, targets are frequently hidden or covered by obstacles or other targets, especially in complex or crowded scenes. Occlusion can be partial or total, and the hidden time of targets is unknown. Straightforwardly, occlusions make detection and tracking tasks extremely challenging and sometimes impossible (e.g., total occlusions). Many methods try connecting two instances of targets (i.e., the moments in which targets disappear and reappear during the occlusion period) based on general movement patterns such as velocity, recorded paths. Using movement patterns to interpret missing paths of individuals is practical in specific surveillance scenes, such as train stations, airports, where the flow of people generally moves from entries to exits. Although applying these types of motion patterns can possibly improve tracking scores in terms of quantity, it seems neglectful with abnormal and unprecedented movements, which are usually seen in criminal activities, for example, a suspect does not move toward any specific destination point. Apparently, these patterns introduce a path model to adapt to maximum people (i.e., inliers) but ignore the small portion whom the security service particularly wants to monitor (i.e., outliers). As the above explications, occlusion problems inevitably cannot be solved in implicit ways, so it remains a challenging problem in detection and tracking.

The follow-up challenge when resolving occlusion problems is how to manage to track multiple targets simultaneously. As individuals usually disappear and reappear multiple times through the entire surveillance video, in order to track those individuals, online tracking algorithms have to relink their appearances every time they reappear on the scene. Therefore, in the online tracking scenario where we consistently and continuously track targets, tracking algorithms are required to pause and resume multiple times corresponding to occlusions. Otherwise, if online tracking is not a requirement, tracking tasks relates to re-identification closely.

The last issue related directly to practical implementations on intelligent cameras is the processing time. For every embedded system, processing tasks must be done within the real-time constraint that is the priority. As we often see in many tracking methods, the accuracy of tracking algorithms is being traded off with their complexity. However, this thesis does not try to cover this issue. In the next section, we clarify all the hypotheses and constraints for our detection and tracking problem with multiple cameras.

1.4 Hypotheses and constraints

Developing intelligent multi-camera video surveillance systems is a multidisciplinary field related to computer vision, pattern recognition, signal processing, communication, embedded computing, and image sensors [194]. Figure 1.3 illustrates the involved disciplines for smart camera networks. Generic multiple object tracking in multiple views for smart camera system is a large and complex problem. Therefore, the more constraints are imposed, the more feasible tracking tasks are.

In this thesis, we mainly focus on the computer vision domain and all related distributed algorithms; hence, we explicitly detail below the list of hypotheses and constraints which define our case study:

- **Camera calibration**, also referred to as *camera resectioning*, is a process of estimating the parameters of a pinhole camera that produces images or videos. These camera parameters are used to correct lens distortion, which are seen captured images, determine image point location in world coordinates, measure the size of the object, or find the camera's position. In general, camera calibration is a task of finding a mathematic model that links the 3-dimensional location of the points of interest in real-world to 2-dimensional image coordinates corresponding to those points. In

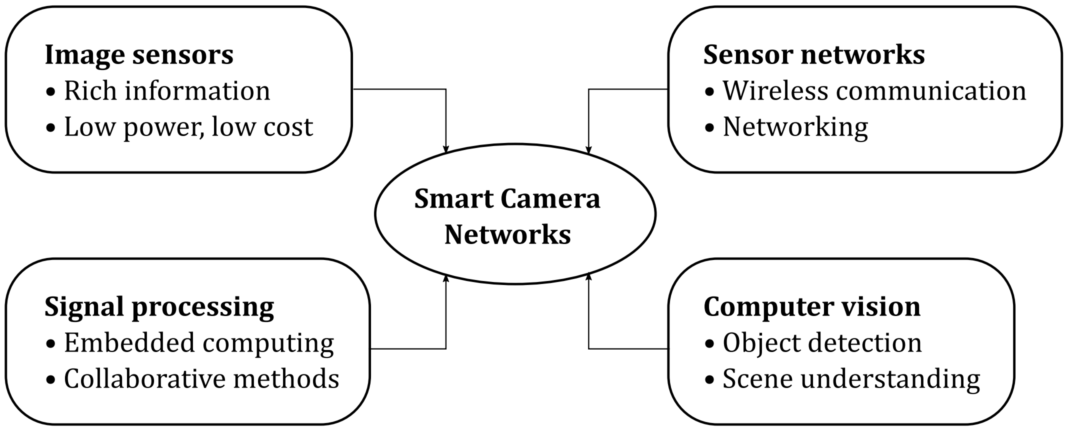


Figure 1.3 – Convergence of disciplines for smart camera networks: image sensors, sensor networks, signal processing for embedded computing, and computer vision [194].

the context of multiple cameras that we are working on, calibrating those cameras is an essential step before making them share their tracking results. In the literature on camera calibration, there are many baseline methods such as Direct linear transformation method, Tsai's method [206], Zhang's method [245]. However, seeing that camera calibration is also one of the main topics in computer vision, this study does not address the camera calibration problem. Therefore, we assume that all calibration information are available to determine the real-world 3D position of every point on the image plane, and vice versa, the 2D image position of every point on the real world.

- **Camera synchronization** is a process of synchronizing the time of capturing images from different cameras. Many computer vision applications require to capture a scene from different points of view, and with a dynamic scene like tracking videos in this study, every frame from cameras needs to be taken at the same time. Synchronizing camera networks involves many synchronizing steps from image-capturing time to the latency of the network. To simplify our case study, we suppose that all cameras are perfectly synchronized.

- **Static camera** is the subject of our study where cameras are fixed at specific locations to film the surveillance areas. In practice, the dynamic cameras (e.g., pan–tilt–zoom (PTZ) camera) are also commonly used in surveillance, and those dynamic cameras involve many other research disciplines such as robotics, control, automation. However, in order to limit our research area and focus on one field, we only consider the static cameras in this study.

- **Overlapping-view cameras** is one of main constraints in our study. As mentioned before, the occlusion problem is still an unanswered question in tracking objects within a single camera. Using multiple cameras observing the same scene but from different angles and positions can technically resolve this issue because when a target is being occluded in one view, it is possibly observable in other views.

- **Unlimited computing capacity** is the hypothesis supposed on each distributed camera what we are working on.
- **Online tracking** and **distributed algorithmn** are our two last constraints. As one of the reasons mentioned above, there is no data storage center required to record all surveillance videos; we are, hence, interested in creating distributed tracking camera systems. This means that all tracking tasks have to be processed online on cameras themselves before collaborating with other cameras in the network.

1.5 Contributions

In this section, we briefly summarize our contribution of this thesis.

- Our first contribution is a robust multi-camera tracking framework for the sparse tracking single view methods by extending the particle filter on a common tracking ground-plane to adapt the multi-view tracking context. The multiple cameras framework allows detecting local appearance variation in a single view and the tracking task switching between cameras in the network. The research result is published in the conference *Reconnaissance des Formes, Image, Apprentissage et Perception* (RFIAP) 2018, Paris, France.
- We adapt an MDP Multiple Object Tracking framework to a multiple overlapping, calibrated, and frame-synchronized camera setting. The problem of associating targets across cameras is modeled, in each frame, by a graph-based approach for which we propose a fast approximate solution. We further exploit multiple views to deal with occlusions and to recover targets' identities, thus to improve the overall identity score. Another essential aspect that we address concerns with the affinity score used in the association step. We propose a robust similarity function consisting of a "trajectory" affinity and a given appearance affinity function that are used to link targets in different views. This research result is published in the *Vision for Interaction and Behaviour undErstanding* (VIBE) workshop of the *British Machine Vision Conference* (BMVC) 2018, Newcastle Upon Type, UK.
- We review several appearance similarities in the state-of-the-art propose a method to combine these appearance features with trajectories as a robust similarity measure for target association across views and finally conduct an exhaustive analysis on the impact of these functions on final results. This research result is presented in the conference ACM/SIGAPP Symposium On Applied Computing 2020, Brno, Czech Republic.
- As another major contribution to the target association problem between different views within an overlapping camera system for online MTMC tracking applications, we reformulate the association problem between two cameras as an Optimal Transport (OT) problem. In order to deal with the potentially varying number of targets, we leverage recent advances in the *unbalanced* formulation of Optimal Transport. Furthermore, we propose to *learn the ground cost* used in the OT formulation. This

learning aims at devising a ground cost that yields the *optimal association accuracy*. It leads to a learning algorithm that evaluates the gradient of the loss function by automatic differentiations through the iterates of an OT solver. We finally adapt the target association framework between two cameras to the context of multiple cameras. This work is presented in a workshop of the 25th International Conference on Pattern Recognition (ICPR), Milan, Italy.

1.6 Plan of thesis

In this section, we present our thesis plan. Indeed, this thesis is generally divided into six chapters: introduction, state-of-the-arts approaches, sparse coding for collaborative tracking mono-object in multi-view, multiple object tracking: target association across multiple cameras, and finally conclusions and perspectives.

- The first chapter, i.e., this chapter, states the motivation of object tracking research and its real-life applications. We mention the context of this thesis, then remind the remaining challenges in visual object tracking. Next, we declare all hypotheses and constraints that we use to conduct this study and briefly list our main contributions. The final part is the outline of the document to show the structure of this dissertation.

- Chapter 2 includes state-of-the-art approaches that are carefully separated into five parts corresponding to the main steps in the tracking process. The first part covers generic object detection methods, which include the classic approaches and the modern approaches based on deep learning techniques with the image datasets. The second part is a brief literature on appearance feature extraction for object tracking. The third part reviews the single object tracking methods in the state-of-the-arts consisting of the classic approaches from the '90s until 2008, the methods using sparse coding for appearance modeling, those benefiting the computing speed from correlation filters and the approaches based on deep learning with the introduction of Siamese networks. The forth and fifth parts present the literature on tracking multiple objects in a single view and multiple views based on either Single Object Tracking methods or Tracking-by-Detection approaches. Finally, the last part presents the benchmarks and performance measures, which are commonly used in the multiple object tracking community.

- Chapter 3 details our contributions to a mono-object sparse coding tracking method based on particle filters for a collaborative and distributed camera network. Our framework for multiple cameras applies the particle filtering strategy for target searching, which is a common tool in Single Object Tracking methods in the literature. Our framework introduces a novel way to detect occlusion events based on sparse coding for further camera collaboration. The tracking results are transferred back and forth between pairs of cameras in the distributed camera network to continuously track their target even with occlusion events happening in some views. The tracking results are compared with the state-of-the-art single view approaches.

- Chapter 4 presents in detail our second contribution, which is an online robust multi-view tracking method for a collaborative and distributed camera network. The chapter is divided into three sections. The first section describes our multiple camera multiple object tracking framework, which supports Single Object Tracking methods to implement to track many targets in the context of multi-view tracking simultaneously. The second section details our data association method to associate targets across cameras based on the appearance and trajectory of targets. Along with trajectories, many appearance features used to measure the similarity between targets across views are studied. Numerous experiments on different configurations with different similarity measures are carried out in comparison with the original single-view approach. The tracking results are validated on the common videos multi-view databases with the standard performance scores, including MOT and ID-measure scores. The third section begins with the problem of combining multiple features to differentiate targets across different cameras. Addressing this problem, we propose the second target association method for pairs of cameras by reformulating it as an Unbalanced Optimal Transport problem. This approach considers associating targets in one view to those in another as finding an optimal transport plan which transports an empirical distribution to another one with a minimum transport cost. The transport ground cost is deduced from a distance between two distributions. We use a deep neural network to encode the trajectory and appearance of targets for learning the metric distance in the OT problem. The novel target association method is well adapted to our multi-camera tracking framework. The experiments are conducted on standard benchmarks to compare the performance of both data association methods.

- Chapter 5 concludes the thesis, discusses the remaining challenges in multi-view multi-target tracking, develops the other perspectives on this tracking problem with new initiatives to tackle problems, finally talks about the current and future applications in the real world.

Chapter 2

State of the Art

2.1 Object detection

Object detection is one of the fundamental problems in computer vision. Its applications widely range from simple tasks such as recognizing and identifying objects to more complex tasks such as tracking mobile objects in autonomous driving or surveillance tracking. The goal of object detection is, given categories, e.g., human, car, pets, to determine whether or not they are in an image and their positions.

2.1.1 Classic methods and image datasets

The history of object detection can be divided into two periods: The first classic approaches from 1999 until 2012, and the machine learning techniques since 2012. The most preliminary challenge in computer vision is to detect objects in images. The object detection research achieved a milestone when Lowe [149] released the Scale Invariant Feature Transform method (SIFT) in 1999, which can find identical objects with the corresponding matches to a given one. This method gives remarkable robustness concerning translation, rotation, illumination, and viewpoint. However, the problem remaining is to match the objects belonging to the same category but not identical, e.g., grouping image of cups which are taken from different cups.

The Lowe's method [149] inspired the object detection community with his invariant keypoints resisting the changes in scale, rotation, viewpoint, and illumination, then the detection trend shifted from finding global appearance features (e.g., shapes, structures, color) to focusing on local descriptors. Since then, finding handcrafted local invariant descriptors had become a popular trend. These local descriptors include Haar features [211], SIFT [148], Histogram of Gradients HOG [55] and Local Binary Pattern (LBP) [164]. After obtaining these features, they are usually concatenated into a vector or transformed into a latent space, such as Bag of Words [131], Deformable Part Model [77], where they can be classified by a supervised method, e.g., Support Vector Machine (SVM) [49]. In 2011, as a remarkable contribution to object localization, the selective search for object recognition by Van de Sande et al. [207] had a strong influence on the later methods. Although object detection algorithms have addressed the *searching object problem* with noticeable results, it remained a question: Can computers detect objects in an image without knowing what is inside? The object detection challenge has become much harder since we required a complete automated object recognition process. This task is called *generic* object detection. Given a large number of images, the challenge of object detection is to determine what is inside and where it is. The following subsection is about deep learning approaches that answered the above question.

Additionally, during this period, PASCAL Visual Object Classes (well-known as PASCAL VOC [74]) was released in 2007 to contribute a large dataset and a fair comparison for all state-of-the-art methods. A couple of years later, Wang et al. [67] also published the first large-scale image dataset in 2009. These enormous image datasets are the prior elements for the advancements of deep learning techniques in the following decades.

2.1.2 Deep learning methods

Machine learning techniques have been developed in the 1990s with few applications due to the limitation of computational power and data. One of the most relevant applications in vision is the digit recognition algorithm with the MNIST handwritten digit database and the birth of Convolutional Neural Network (CNN) by LeCun et al. [135] in 1998. Since then, there were very few machine learning approaches addressing computer vision problems until 2012 when Krizhevsky et al. [127] released their method with their GPU implementation. They built a neural network structure using multiple convolutional blocks, trained the entire neural network on GPU, and gave a breaking performance on imageNet [67]. By taking advantage of the parallel computational power on GPU, the implementation was able to train a bigger CNN structure from a massive training data in a reasonable time. Since AlexNet [127] was the board leader of ImageNet Challenge in 2012, deep learning techniques received lots of attention from the community, and it has emerged as a promising method for a powerful feature representation with an end-to-end learning approach. Its accuracy crucially relies on the volume of training dataset and the depth of the neural network [99]. Because of the availability of large scale image datasets such as ImageNet[67], PASCAL VOC[74], MS COCO[143], the accuracy of deep learning object detection algorithms dramatically increased and overpassed human performance [145]. According to the neural network architectures, the state-of-the-art deep learning object detection methods can be separated into two categories [145]: two-stage detection approaches and one-stage detection approaches.

2.1.2.1 Two-stage methods

The main idea of this type of method is to pre-process images to propose the Region of Interest (RoI) and then recognize the object from wrapped image patches extracted from the RoI. Indeed, the pre-processing step primarily aims to localize objects, and then the main detection step labels these given regions by a specific classifier. Following this detection scheme, Girshick et al. [88] proposed the Region-based Convolutional Neural Network (RCNN) consisting of one region proposal step via Selective Search [207] and one neural network to label each region proposals. Notice that the Neural Network includes: the convolutional blocks serving as a *feature extractor* and a single/multi-class *classifier* such as C-SVM. Image patches used as an input of the CNN are cropped from proposal regions and rescaled to have the same size. CNN models are pretrained using multiple image datasets such as ImageNet [67], PASCAL VOC[74] or MS COCO[143].

Observing an inconvenience that the RCNN [88] only accepts an unique-size image input while the CNN block can get the arbitrary size, He et al. [98] inserted an SPP (Spatial Pyramid Pooling) layer, which aims to obtain fixed-length features, between the CNN block and the Fully Connected (FC) layers. The most significant disadvantage of these above methods is that detecting multiple objects in a single image means we need to feed the network multiple times. Repeating this process slows down the algorithm, and it is not an effective way to detect many objects, because in most cases, the objects are usually superimposed on each other, and theROIs share the same regions. In 2015, Girshick [87] introduced the Fast RCNN, which applies the Selective Search at the last CNN layer to extract features of RoI, called *feature map*, before entering to the FC layers. This manner helps the algorithm benefit from the sharing computation of convolution since the input of Fast RCNN is an arbitrary-size image comparing with multiple extracted regions in the older version RCNN [88]. Meanwhile, instead of using the Selective Search [207] to obtain RoI, the Faster RCNN proposed by Ren et al. [176] uses a Regional Proposal Network (RPN) which takes the feature map as the inputs (i.e., the output of the last CNN layer) for each spatial location, e.g., objectness classification, bounding box regressor. The RPN helps the network run faster in terms of magnitude [176] and retain sufficient geometric information for accurate object detection [136] as well.

Inspired by Fully Convolutional Network (FCN) for semantic segmentation [147], Dai et al. [54] introduced the Region-based Fully Convolutional Network (RFCN), which helps to minimize the amount of computation that cannot be shared. Indeed, the FC layers in *classification* are pinned to the head of the feature maps and *convolutionalized* [147] to be able to accept more than a fixed-size image input, then the feature maps go through these FC layers to generate *position-sensitive score maps*. Finally, each RoI extracted from these maps gets through a multi-class classifier for the type of object and a Bounding Box regressor for the final localization. Some extensions of RCNN such as Mask RCNN [97], Light Head RCNN [142] significantly improved the speed and accuracy of RCNN for object detection. Recently, to improve the accuracy of bounding box localization, Jiang et al. [113] proposed IoU-Net as an alternative optimization-based bounding box refinement for other deep object detection methods. Figures 2.1 summarizes the development of two-stage deep learning methods of object detection.

2.1.2.2 One-stage methods

Unlike the region-based (or *two-stage*) strategy, this approach uses one CNN architecture to localize objects' positions via a bounding box offset and predict class probabilities without any additional *(*region proposal) searching step. This setting only needs a single feed forward CNN network to obtain results directly. Therefore, this elegant and straightforward end-to-end detection pipeline can be directly optimized on detection performance during the training phase. The very first work implementing this idea is the DetectorNet by Szegedy et al. [197] in 2013. In detail, their idea to localize objects on an image is to quantize input image into multiple cells of a fixed-size grid, called *coarse grid*, and then

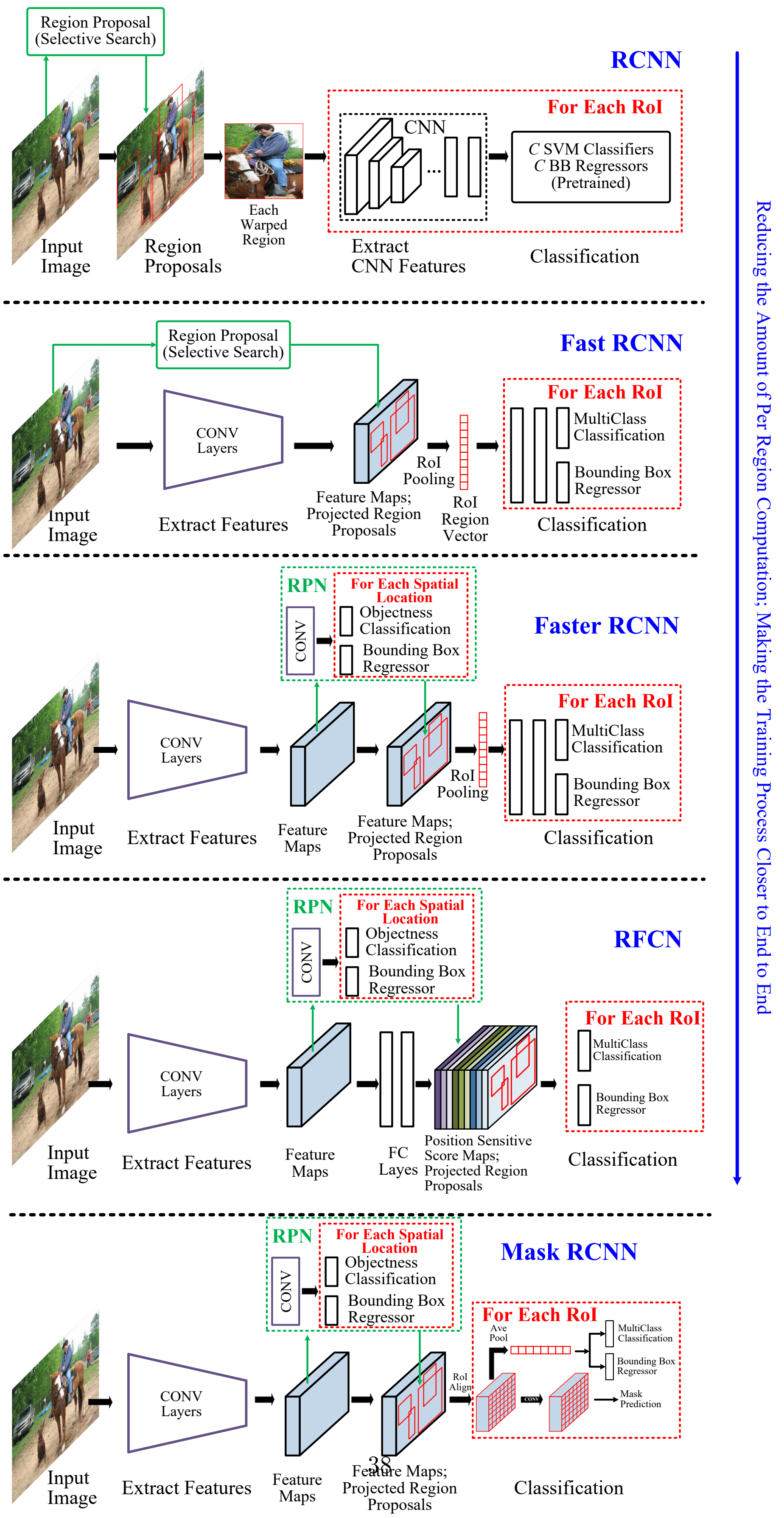


Figure 2.1 – Summary of the two-stage deep learning methods in object detection[145].

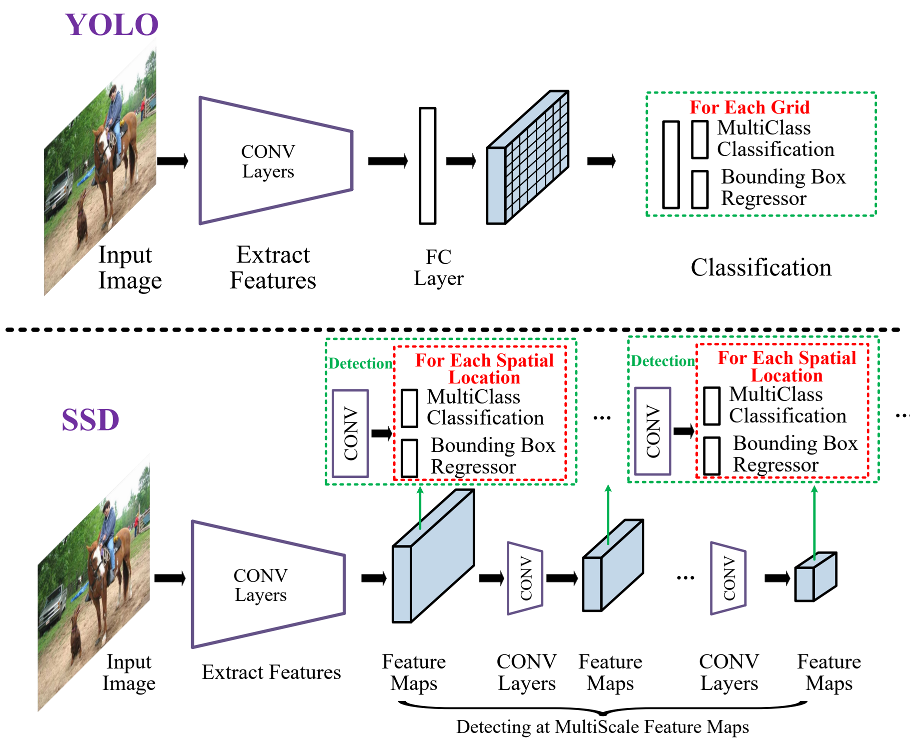


Figure 2.2 – Summary of one-stage deep dearning approaches in object detection [145].

determine which cells contain objects, i.e., *foreground* and which does not, i.e., *background*. They used AlexNet [127] with a regressor on the top of the prediction masks, which are, by definition, the cells that mostly overlap with the bounding boxes of objects. Accordingly, these masks are fed to a series of CNNs to obtain the object's bounding box. The disadvantage of this method is that this procedure repeats multiple times at different scales to detect objects of different sizes. Meanwhile, Sermannet et al. [188] proposed OverFeat with a deep convolution network performing object detection in a multi-scale window sliding. Practically, in the training phase, the CNN is applied on an image patch and produces one single output, while in the testing phase, applying on a larger image results in a feature map. They naturally share computation between one-pixel sliding windows. As the output of the classifier, the feature map, which indicates on each pixel a class and its confidence score, enters into a regression network to predict the bounding box. This method is advantageous in terms of speed, but trades off accuracy compared with RCNN [88].

Having a similar idea of DetectorNet [197] to localize objects on the *coarse grid* of the input image, Redmon et al. [173] introduced YOLO (You Only Look Once) which is a simpler model. YOLO possesses a novel CNN structure that unifies object classification and bounding box regression within a single architecture. The method directly predicts objects from candidate regions or *grid cells*, which is actually equivalent to the *marks* in the DetectorNet paper [197]. In detail, YOLO divides input images into a $S \times S$ grid, each cell predicts *only one* object via a fixed number of different shape bounding boxes containing box location {x-offset, y-offset, width, height}, B *box confidence scores* and C *conditional class probabilites*. Hence, YOLO's output has a shape $(S, S, B \times 5 + C)$. Unlike

OverFeat [188], the searching space of YOLO is greatly smaller, explicitly $S \times S \times B \times 5$. Eliminating region proposal step and having a limited space search, YOLO can run very fast and exceed far the real-time requirement. As a setback of this method, the accuracy of YOLO is relatively low, it sometimes fails because objects are too small or there are so many of them due to the fixed-size grid. Later, Redmon and Farhadi [174] released YOLOv2, which achieved state-of-the-art performance with 9000 object categories (YOLO9000) while still running in real-time. The illustration of YOLO approach is shown in Figure 2.3

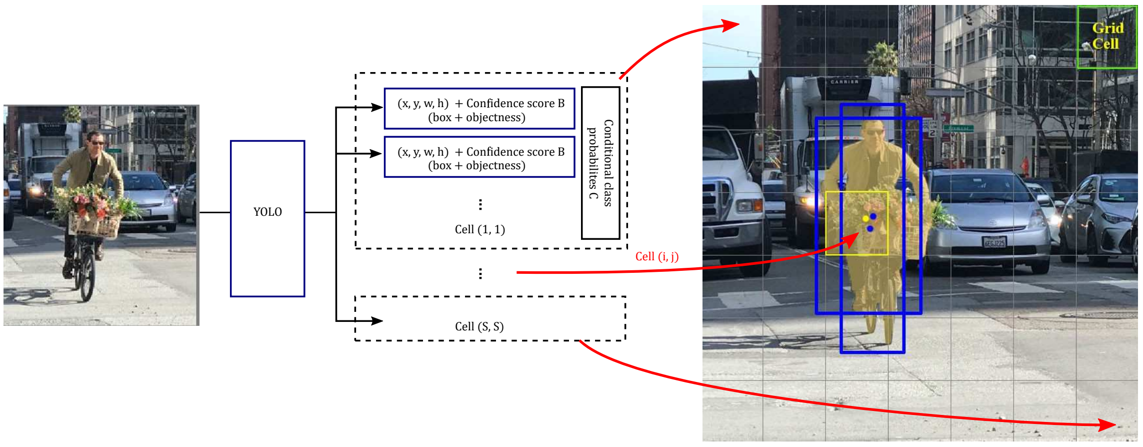

YOLO
(x, y, w, h) + Confidence score B
(box + objectness)
(x, y, w, h) + Confidence score B
(box + objectness)
Conditional class probabilites C
Cell (1, 1)
Cell (i, j)
Cell (S, S)
Grid Cell

Figure 2.3 – Illustration of YOLO detection object approach [197].

Following the single-stage detection scheme, Liu et al. [146] introduced Single Shot Detector (SSD), which efficiently combines the idea of RPN (Regional Proposal Network) from Faster RCNN and cascaded CNN blocks to obtain multi-scale feature maps. Indeed, the SDD consists of a common feature extracting CNN network, such as VGG [193] followed by a series of CNN networks. Accordingly, the feature map from VGG goes through many multi-scale cascading CNN blocks. After going through each CNN block, the feature maps gradually decreases in size and detail of information. Thus, the first CNN blocks detect small objects, while others detect for the larger ones. In practice, SDD has comparable accuracy with the state-of-the-art detectors such as RCNN [88] or Faster RCNN [176], while having a higher speed in comparison with YOLO [173]. Figure 2.2 summarizes the one-stage deep learning approaches in object detection.

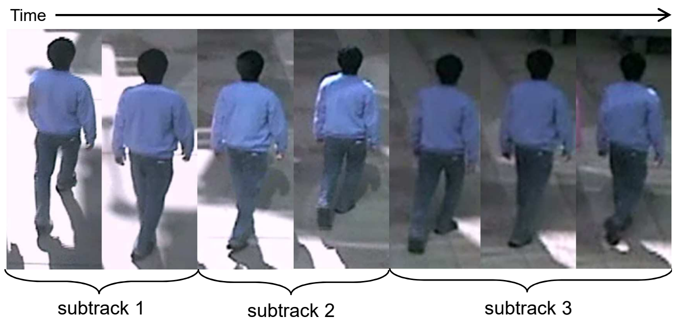


Figure 2.4 – The challenges while observing a target from different views: the width-height ratio of a standard bounding box is fixed while this ratio can be changed in a different view like from high altitude; the lighting condition effecting to the color pattern inside bounding box of target [40].

2.2 Appearance feature extraction for object tracking

In pattern recognition, extracting feature is a crucial step, which aims to measure the similarity or dissimilarity between known things especially in object detection, object tracking or re-identification. As mentioned in the previous section, object detectors use appearance features to classify an object into known categories. For object tracking task or human re-identification task with multiple objects belonging to same categories, appearance feature is used to discriminate an object instance from other objects in the same class. Within most surveillance applications, people and vehicles are the two main targets. There are several significant challenges to identify these targets among multiple detections. The first challenge is to deal with the mix of targets and the background, due to the rectangular shape of detections, i.e., bounding boxes. For the detections of vehicles, the rectangular shape well adapts to the car shape, the background zone inside of the box is relatively minor and negligible, but it is not well-shaped to adapt articulating objects like human body.

In this section, we mainly focus on the most common appearance features used to track people in the literature. Firstly, as many detectors do not release well-cropped bounding boxes, which might contain the background rather than its target, in reality, around 50% to 60% its surface is the background, the remaining is the body parts. Secondly, a single target can have many different poses, e.g., the front, behind, side, or even top. Lastly, the lighting condition has a huge impact on the appearance consistency of targets, especially color. This is caused by either weather conditions or different color sensors of different cameras. The illustration of these challenges is depicted in Figure 2.4.

The appearance of a human can be described by various patterns, but the most common one is color that is widely used in many papers (Cai and Medioni [34], Chen et al. [39, 41, 40], Das et al. [62], Gilbert and Bowden [86], Javed et al. [110], Jiuqing and Li [214], Kuo et al. [129], Zhang et al. [239, 240]). The color features can take many forms, but the most useful and simplest one is the color histogram with different color models such as RGB (Red, Green, Blue), CMYK (Cyan, Magenta, Yellow, Black), HSL (Hue, Saturation, Lightness) and HSV (Hue, Saturation, Value). The texture of targets is also a good indicator (Cai

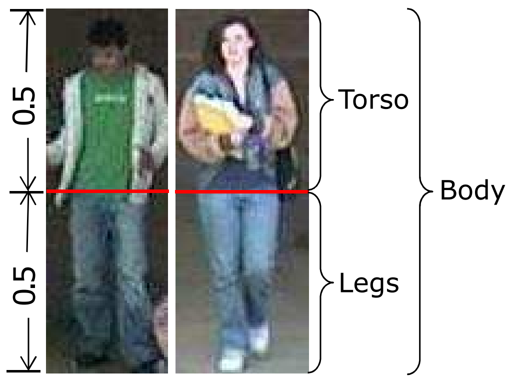


Figure 2.5 – Human body being encoded separately into two main parts: torso and legs [40]

and Medioni [34]; Chen et al. [40]; Daliyot and Netanyahu [56]; Kuo et al. [129]; Zhang et al. [239, 240]).

Lighting variations or illumination changes are mitigated by enhancing the color histogram with normalization (Cai and Medioni [34]), using exemplar-based approaches (Chen et al. [40]), or Brightness Transfer Functions (BTF) learned with (Das et al. [62], Javed et al. [110]) or without supervision (Chen et al. [39], Gilbert and Bowden [86], Zhang et al. [239, 240]). The histogram of oriented gradients (HOG) [55], which can resist to lighting variations, is also a common features to extract the appearance of objects.

Furthermore, to improve the appearance model, human parts can be encoded separately, such as torso versus legs (see in Figure 2.5). Meanwhile, in the aspect of a non-rigid object type, moving people have their limbs gradually articulating around their main body parts through successive frames, Deformable Part Model (DPM) introduced by Felzenszwalb et al. [77] are commonly used to model human body in tracking videos. There are also other attempts to combine both human body detector with facial detectors [37, 103]. Proposing a generic object descriptor, Choi [45] introduced Near-Online Multi-target Tracking (NOMT) approach with the Aggregated Local Flow Descriptor (ALFD) that aggregates multiple local Interest Point Trajectories (IPTs) which, by definition, are the matching points found by local interest point detectors between two detections via optical flow algorithms [28]. The author also presented a hierarchical affinity measures based on IPTs between two detections. By incorporating the ALFD with a motion/appearance model, the tracker has the ability to run in real-time with high accuracy. However, the tracking results are delayed a certain frames due to the Multiple Hypotheses Tracking (MHT) process, which is used to determine the best hypotheses (trajectories).

Besides the hand-crafted appearance features, many papers [75, 154, 155, 182, 243] deployed sparse coding techniques for target's appearance representation. Mei et al. [154] first introduced spare coding for appearance representation in vehicle tracking. There are many works [155, 241, 144, 112, 12, 250, 217, 182, 243, 233, 215, 242, 238] following up to enchance the ability to represent targets' appearance changes in videos. Extending spare coding in multiple tracking object cases, Fagot-Bouquet et al. [75] proposed a formulation of the multi-frame data association step as an energy minimization problem with an optimization energy that efficiently exploits sparse representations of all detections.

The enhancement of appearance models is also achieved by reinforcing the discriminativeness of the selected features. Martinel et al. [152] proposed using saliency information. Zhao et al. [246], Cai and Medioni [34], Chen et al. [40], Daliyot and Netanyahu [56], Das et al. [62], Jiuqing and Li [214], Kuo et al. [129] learn specific features on body parts or in the image such as Bedagkar-Gala and Shah [17, 18], Cheng et al. [43] Meanwhile, several papers encode the appearance features on an articulated (Baltieri et al. [9], Cheng and Cristani [42]) or monolithic (Baltieri et al. [10]) 3D body model. Long Short-Term Memory (LSTM) combining with the extracted features from CNN by Sadeghian et al. [184] is used to adapt the evolving body parts while discriminating the background during videos.

The appearance features have been studied intensively in the re-identification community. They focus on how to distinguish different people in a collection of detections. The recent literature in person re-identification relies mainly on deep learning techniques, which has been showing their simplicity and high performance in visual representation this decade.

2.3 Single-object tracking

Visual object tracking is one of the long-standing research fields in computer vision. Primarily, tracking a single object is the most fundamental task in visual tracking. Therefore, in this section, we discuss the classic and modern single object tracking methods of the state-of-the-art. The history of the development of single object trackers is depicted in Figure 2.6

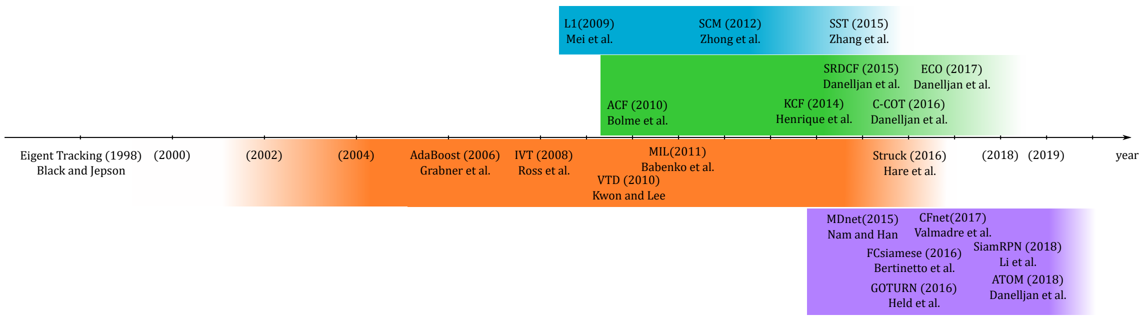


Figure 2.6 – Development of Single Object Tracking methods. There are four main approaches in Single Object Tracking: classic (orange), sparse-coding based (blue), correlation filter (green), and deep learning (violet) approaches.

2.3.1 Classic methods

Most visual object tracking approaches are generally presented in multiple steps, which are shown in Figure 2.7. The scheme of visual object tracking approaches consists of finding suitable appearance representations for objects of interest, searching targets in the successive images, estimating object's positions, and updating appearance models.

A variety of object modeling approaches have been proposed due to different types of

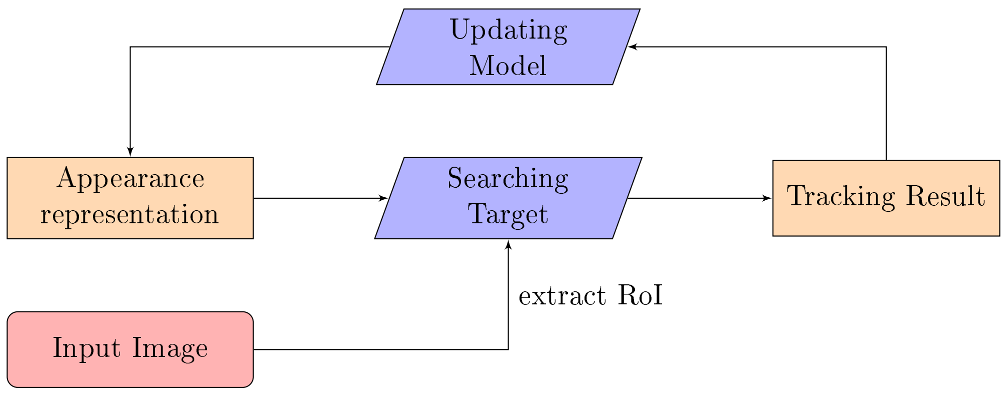


Figure 2.7 – General scheme of Single Object Tracking approach

objects, which are mainly divided into two classes: rigid and non-rigid (or *articulated*) objects. Rigid-body objects generally do not have any deformation during their movements, concisely, their shapes remain unchanged while moving, but their visual appearance can change in different points of views. In practice, tracking applications are interested in rigid-body objects like vehicles such as cars, airplanes, boats. On the contrary, articulated objects are more complex and harder for modeling as their sub-body parts move around their main body during movements. This causes their appearance to change frequently in video sequences. In a survey of Yilmaz et al. [234], the classic object representation includes keypoints, centroid, skeleton model, bounding box, object contour, and object silhouette. Keypoints, centroid, and primitive shapes, polygons are commonly deployed to track rigid objects which have a particular shape or texture. Meanwhile, object contour, silhouette, and skeleton are used for non-rigid objects such as pedestrians, animals. Object representation is described via appearance extracted from the bounding box. These appearance representations consist of: appearance features, e.g., color, textures, templates, e.g., image patches of the objects, active appearance models, e.g., landmarks on the boundary and inside objects, multi-view appearance models which can encode object's appearance from different views. In most of the state-of-the-art approaches, bounding box is the simplest and most commonly used among these above object models. Due to the popularity of the bounding box in tracking, the appearance model of object tracking is usually replaced by the term “appearance features”, which are the visual features extracted inside the bounding box containing the object. Indeed, appearance features intensify the uniqueness of objects through the whole video.

Initially, in classic tracking approaches [234], several common features are deployed such as *color* which is decomposed into three channels RGB in image processing, or sometimes HSV (Hue, Saturation, Value); *optical flow* which is a dense displacement field of keypoints which depicts the motion of the objects (e.g., Lucas and Kanade (LK) [150]); *texture* which is the visual variation on the surface of the object. However, the LK approach does not consider the entire appearance of targets, which leads to the poor performance of those trackers. In practice, objects' appearance frequently changes during the tracking process, specifically in the case of non-rigid objects, due to the change of camera's POV, or the illumination on the scene and so on. Multiple learning-based methods have been proposed to adapt to the object's appearance variations efficiently. In the paper of Ross et al. [181], the appearance of objects is represented by a low-dimensional subspace, and this representation is incrementally updated every frame. This method showed its capacity to adapt to the variation of the target's appearance, but its drawback is that its intensive updating causes the *drift problem*, which is observed when the appearance templates drift out of the target after a long enough tracking time [153]. Since updated image patches might contain tiny translation errors, these errors are accumulated through a time period and become significant finally. To cope with this issue, Grabner et al. [90] proposed a discriminative approach (OAB-Online AdaBoost tracker), which formulates the initial tracking problem as a classification problem. A set of candidates is sampled around the target's position obtained from the last frame. Then they are divided into two groups: positive samples and negative samples. The classifiers are learned to distinguish the target (positive samples) from the background (negative samples). However, this method cannot help the tracker altogether avoid updating the templates obtained from inaccurate tracking results. To address the drift problem, Grabner et al. [91] introduced the semi-supervised boosting method, which combines a given prior and an online classifier. In the Multiple Instance Learning (MIL) method by Babenko et al. [7], ambiguous positive and negative samples are put into two bags (i.e., *bag-of-word*) to learn a discriminative classifier. The intuition behind is that the positive samples are generated relatively close to the target's center, while the negative ones are barred from the center with a fixed distance. This keeps the tracker from updating the samples, which are too close to the *barrier* of negative samples. Zhong et al. [250] used this type of discriminative classifiers and combined it with a generative model. A common point of all these algorithms is to provide an appearance model that has both the discriminative ability and robustness; however, balancing between those goals seems problematic.

In terms of localizing objects, object detection also aims to find the location of the object of interest in images, but the searching mechanism makes tracking algorithms different. Tracking algorithms would rather search its target in the Region of Interest (RoI) based on its last position than scanning on the entire image. According to the survey of Yilmaz et al. [234], object tracking methods or *search methods* are divided into 3 categories: point tracking, Kernel tracking and silhouette tracking. The first catergory is separated into deterministic methods [210, 186] and statical methods such as Kalman filter [30], Joint Probalistic Data Association Filter (JPDAF) [14, 172], particle filter [109]. Having seen the particle filter as the most common technique in many Single Object Tracking approaches, as well as ours, we will detail this technique in Sec. 3.1. In kernel tracking approaches, there are several relevant papers based on appearance models, e.g. Mean-shift [48], KLT [190], Eigentracking [25]. Recently, many trackers have risen by applying dense searching approaches while resulting in impressive performance such as correlation filter trackers [26, 102], which will be discussed in the next section.

Besides designing a good appearance model, another efficient way to handle appearance variation is to update the appearance model frequently. Initially, the update mechanism is developed to combine the reference template of the target at the beginning of the tracking process with the most recent tracking results [153]. Since then, there have been other different update approaches depending on the appearance model of trackers such as online boosting [90], online mixture model [111], and incremental subspace update [181]. With the discriminative model, such as [7, 95], the trackers update the appearance model by training an online classifier as supervised learning every frame. Meanwhile, the methods in [91, 115] reformulated the online supervised classifier update as semi-unsupervised learning.

In terms of object detection at the first frame, this step is called "Initialization" (or "re-initialization" for reestablishing the tracking process when failing). This task can be done manually or automatically. Regarding to the automatic mode, moving object detection can be addressed by generic object detection which has been described in the previous section. In the case of the invariant and simple scene, objects can be extracted by background subtraction approaches [81], segmentation approaches [32, 158] or SVM, e.g. Adaboost framework by Viola et al. [212]. There was an exhautive survey on the Single Object Tracking methods conducted by Wu et al. [228] in 2013. In the next section, we detail an efficient approach to represent the appearance of objects via sparse representation.

2.3.2 Sparse coding based methods

Sparse coding has shown its efficiency in visual representation [72] and attracted attention from the computer vision community. In this section, we discuss the sparse coding techniques used in computer vision, especially visual tracking. In principle, sparse coding is a method of signal representation. A signal can be represented as a linear combination of a large number of different signals with the same dimension, which form a redundant matrix. Since the number of columns is much higher than row's, the linear combination of the basic signals in the redundant matrix representing the original signal can be presented by an infinity number of coefficient vectors, i.e., the underdetermined system. With reference to sparse coding, only the coefficient vector, which has the least non-zero elements, is considered as the solution, the original signal now is being represented by a few signals in the redundant matrix. For applications, sparse coding is commonly used in image processing [151], such as denoising, deblurring, resolution increasing. In computer vision applications, the first remarkable work using sparse coding is the facial recognition algorithm by Wright et al. [226, 225]. Since then, the method has been increasing its influence on many applications of computer vision, including visual tracking.

According to the survey conducted by Zhang et al. [238] in 2013, sparse-coding-based tracking algorithms have two major contributions: first, Appearance Modeling based on Sparse Coding (AMSC) and secondly, Target Searching based on Sparse Representation (TSSR). In visual tracking, the appearance model of a target is represented by image patches, which are stored in a redundant matrix, called *dictionary*. Only a subset of those image patches are selected to encode the most dominant features of the target. On the other hand, Target Searching based on Sparse Representation intends to find the best coefficients to represent the given candidates in a dictionary. The pioneer works using sparse representations in visual tracking have been introduced within the particle filter framework [70] by Mei et al. [154, 155] and Zhang et al. [241]. Each candidate is represented as a sparse linear combination of target and trivial templates. This set of templates, i.e., dictionary, is updated regularly in order to maintain an up-to-date appearance model. The role of the trivial templates is to account for possible object occlusions (see Figure 2.8). The representation coefficients for each candidate are obtained by solving an ℓ_1-penalized least squares minimization problem.

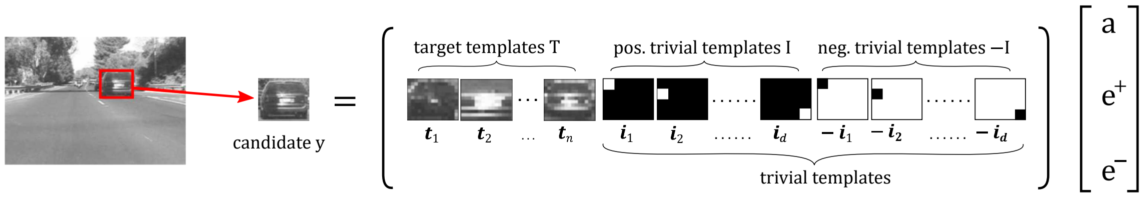


Figure 2.8 – Sparse representation with target and trivial templates handling occlusion [155].

Inspired by the method in [154], there were many papers improving the robustness of the approach and reducing the computational cost. Liu et al. [144] introduced a two-stage sparse coding to jointly minimize the target reconstruction error and magnify the discriminative ability. Mei et al. [156] proposed a modified particle filter framework that removes insignificant samples before encoding the whole samples in the dictionary. Bao et al. [12] introduced a minimization model in which the method combines a ℓ_1-norm and a ℓ_2-norm to improve accuracy, and used a fast solver FISTA [16] that allows the algorithm to perform in real-time.

The important works [112, 250] in 2012 contributed to more effectively representing target and adapting to appearance variations. Jia et al. [112] proposed a structural local sparse appearance model to avoid drift problems and to handle partial occlusions. Concretely, the representation of targets is a set of overlapping local image patches covering the entire target's region, and then candidates are encoded in a dictionary containing local image patches of the target by the representing coefficients. Previously, these coefficients are pooled with respect to the position of their corresponding image patches in the structural local appearance model described before, as called *alignment-pooling* step. The pooled features show which patches of candidates belonging to the target and where the target's centroid locates at inside the bounding box. Zhong et al. [250] presented a new appearance model combining a Sparsity-based Discriminative Classifier (SDC) and a sparsity-based generative model (SGM) to adapt to the appearance change itself (*generativeness*) while reinforcing the distinctness of the target with the background (*discriminativeness*). In brief, discriminative features are sparsely selected by encoding a dictionary consisting of positive and negative samples generated around the target. After selecting features, the reconstruction of the candidate based on these features focuses on representing the target more genuinely and effectively. The reconstruction error is later combined with the histogram of sparse coefficients, which are encoded in a dictionary of k-means clusters of the local patches covering all over the target at the first frame. By incorporating SDC and SGM, the method can enhance the accuracy of the tracker and have better dealing with occlusions.

Wang et al. [217] proposed an online dictionary learning algorithm for updating the object's templates, reformulated the sparse coding problem with the Huber loss. The loss function allows the sparse coding problem to remain equivalent to the standard approaches [154, 156] while eliminating trivial templates. This helps the dictionary enormously reduce its size, so the tracker benefits the computational cost.

Wang et al. [215] exploited both classic Principal Component Analysis and sparse representation for efficiently learning dictionaries. Zhang et al. [242] introduced the consistent low-rank sparse representation to obtain a structured dictionary that allows candidates to be computed jointly and efficiently. Yang et al. [233] provided a framework of an online discriminative dictionary learning, meanwhile, Rousseau et al. [182] proposed a dictionary learning approach to model the appearance of targets, which results in a smaller size of the dictionary. Zhang et al. [243] released the Structural Sparse Tracking approach, which generalizes all the structural sparse appearance models. This method uses a predefined *spatial layout* to sample local image patches inside candidates and represents these patches in a local dictionary of image patches. For each candidate, by following the spatial layout structure, the sparse coefficients are rearranged to reconstitute the coefficient of the entire candidate as in global sparse modeling approaches. As a result, the method incorporates candidates with their local patches to jointly represent the target without losing the defined spatial structure of the model. In 2013, Zhang et al. [238] made a survey of sparse coding based tracking methods and conducted an experimental comparison of these trackers.

2.3.3 Correlation Filter methods

Searching the target in the area near to the last position of objects makes tracking problem different from detection problem. Object searching is to find the best matching candidate among those generated around the last object's position. Most classic trackers adopt particle filter frameworks that use a Monte Carlo approach to represent the target's position via a probability density of particles. The advantages are the convenience of estimating and propagating the posterior probability density through frames; dynamic model (or *state transition model*) and observation model can be changed to adapt to different tracking methods and appearance models. Nevertheless, the disadvantage of these frameworks is that in order to ensure the accuracy of the tracking process, the number of particles dramatically is increased according to the target's state dimension. There were many works, previously mentioned [156, 112, 242, 243], addressing this critical issue.

On the contrary, the first correlation filter in visual tracking was introduced by Bolme et al. [26] in 2010 to tackle the candidate searching problem differently. Instead of generating a large number of samples, the method densely convolves the correlation filter, e.g., target's templates, with the search area. Then, the most expensive computation, which is the *correlation*, is efficiently and quickly computed in the Fourier domain. In this study, to avoid confusion, we use the terms the *spatial domain* for "time domain" and *Fourier domain* for "frequency domain". Computing the solution in *spatial domain* is highly expensive, so the biggest advantage of this approach is the capability to operate the entire calculation in the Fourier domain, which makes these trackers overperform others in speed. Figure 2.9 depicts the general scheme of correlation filter tracking approaches.

In the correlation filter (CF) based methods, the tracking problem is formulated as a regression problem. Instead of sampling training candidates around a target, a circular matrix (as illustrated in Figure 2.10) is used to create an artifact of the movement of the target as shown in Figure 2.11 which is an illustration of the convolution of an image patch containing the target's template and the search image.

In detail, given two same-size vectors u and v, the multiplication of the circulant matrix of the vector u and the vector v is equivalent to the correlation of two vectors u and v.

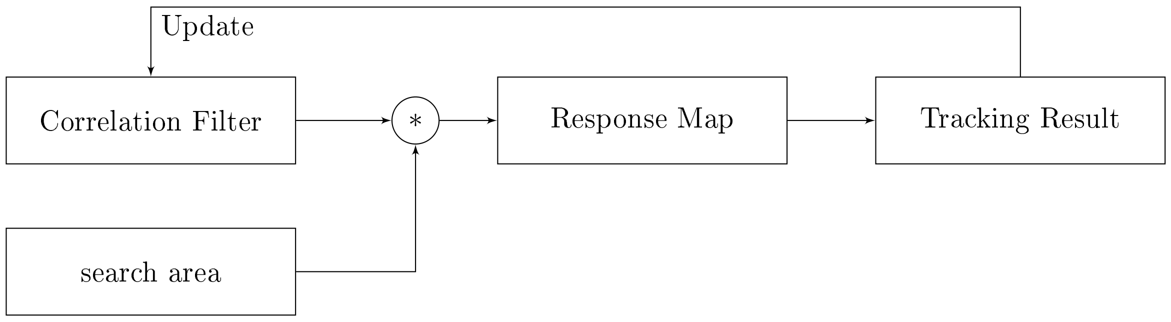


Figure 2.9 – General scheme of Correlation Filter tracking approaches.

Furthermore, this correlation in the spatial domain can be operated much more efficiently by an element-wise multiplication in the Fourier domain. As a result, the computation cost of correlation filters is very low, so these types of trackers usually outweigh other types in terms of speed. Henriques et al. [102] evolved the initial linear regression into the non-linear regression by using the kernel trick [107]. Their kernel matrices possess a circulant structure (see [101] for further details) that allows the correlation in their tracker to be calculated via the wise-multiplication in the Fourier domain.

Regarding the update mechanism of CF trackers, which makes them resistant to the appearance variation of their target, some simple methods are implemented, such as updating the filter with a small coefficient every frame [102]. Danelljan et al. [60] proposed the Spatially Regularized Discriminative Correlation Filter (SRDCF), which uses the weighted window applying to the filter f in order to penalize the filter coefficients corresponding to background part. The illustration of this method is displayed in Figure 2.12 where the regularization weights penalize filter values corresponding to features in the background. This increases the discriminative power of the learned model, by emphasizing the appearance information within the target's proposed region (the green box in Fig. 2.12) [60]. The weighted coefficients of the filter make the response within the bounding box more discriminative in comparison to the background. To enhance the capability to capture the characteristics of objects, [59] deploys a CNN network such as imagenet-vgg-2048 [38] to extract robust visual tracking features. One of the setbacks of the CF method is the restriction on image resolution that limits the accuracy of the tracking results on the size of bounding boxes. Concretely, the tracking result is obtained from the position of the pixel corresponding to the highest filter response. To address this problem, Danelljan et

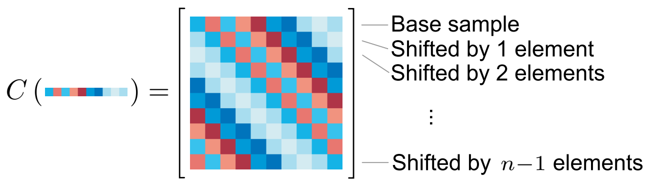


Figure 2.10 – Illustration of a circulant matrix. The rows are cyclic shifts of a vector image, or its translations in 1D. The same properties carry over to circulant matrices containing 2D images.[102]

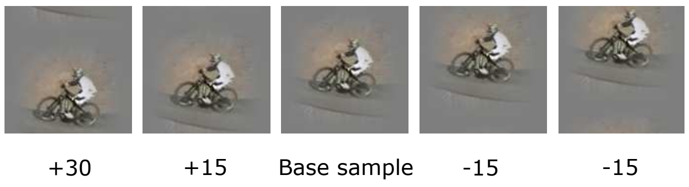


Figure 2.11 – Example of vertical cyclic shifts of a base sample. The formulation in the Fourier domain allows the tracker to be trained with *all* possible cyclic shifts of a base sample, both vertical and horizontal, without iterating them explicitly. Artifacts from the wrapped-around edges can be seen (top of the left-most image), but are mitigated by the cosine window and padding. [102]

al. [61] introduced the Continuous Convolution Operator for visual tracking by using an interpolation function. Convolving the filter with an interpolated input image results in a smooth response and, consequently, a better accuracy. Based on the C-COT (Continuous - Convolution Operator Tracker) [61], ECO (Efficient Convolution Operator) tracker [58] proposed using a Gaussian mixture to model the training data in order to avoid over-fitting caused by recent samples. One of the main obstacles of the basic CFs in object tracking is to adapt to the object's evolving shapes due to the articulation of the object or the change of the angle between the object and the camera while moving. Therefore, the recent CF tracking methods mainly rely on deep neural nets to enhance their performance in feature extraction for object tracking, as the heart of the Siamese structures, which will be described in the next section.

2.3.4 Deep Learning methods

Since the success of deep learning in computer vision, many research works have applied this technique to tracking problems. Notably, Convolutional Neural Networks (CNNs) greatly contributed to representing visual data by showing their outstanding performance in a variety of problems in computer vision. Hence, as one of the first attempts to use deep learning in visual object tracking, Nam and Han [162] presented the Multi-Domain

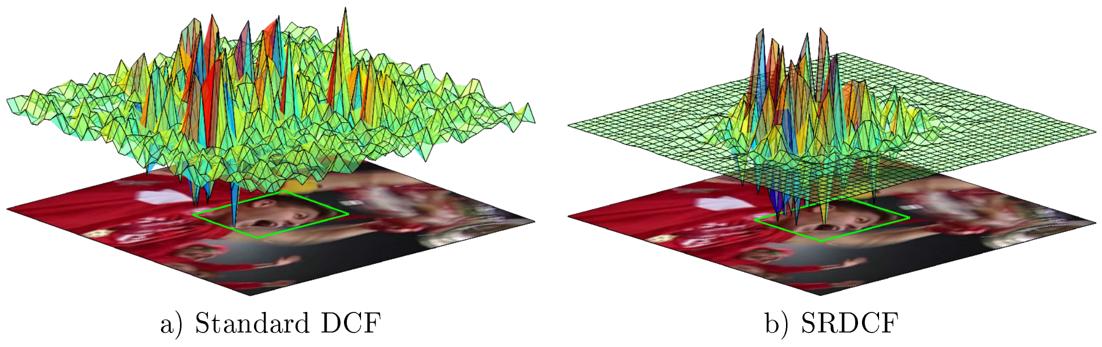


Figure 2.12 – Visualization of the filter coefficients learned using the standard DCF (a) and SRDCF approach (b)[60]

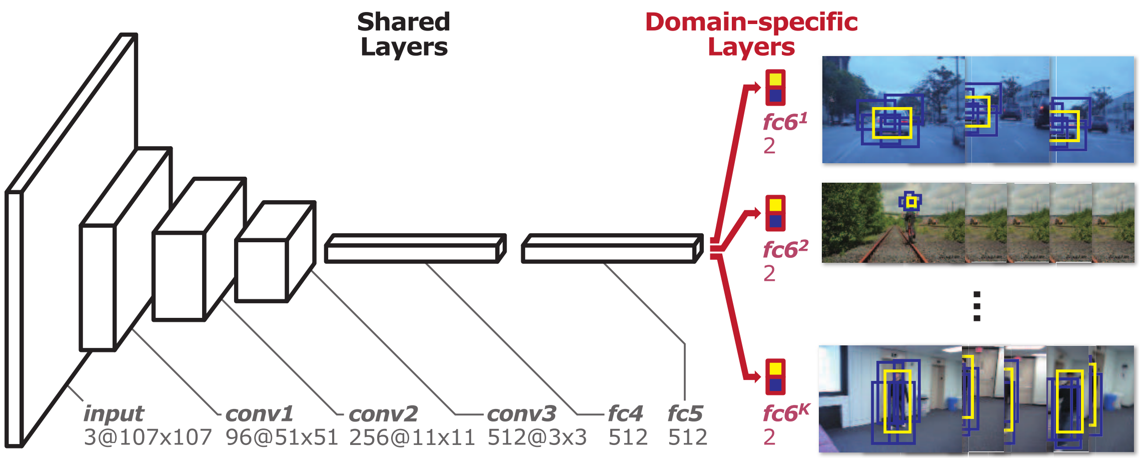


Figure 2.13 – The MDnet architecture, which consists of shared layers and K branches of domain-specific layers. Yellow and blue bounding boxes denote the positive and negative samples in each domain, respectively [162]

Network (MDnet). This method is technically classified as a multi-domain learning algorithm whose training data is considered as a dataset with multiple domains. Each domain is a single metadata attribute, and in the paper, these domains correspond to individual training sequences. In detail, their network consists of a series of shared layers with a number of branches corresponding to specific domain layers, which are simply the binary classifiers. The MDnet's architecture is depicted in Figure 2.13. The authors separate the *head* branches containing the domain-specific information from the shared layers storing the domain-independent information. Indeed, during the training phase, their CNN is trained with a series of single sequences (with a corresponding domain-specific layer for each sequence) and retrained multiple times in the same order. Then during the testing phase, all the pretrained domain-specific layers are replaced by new ones. Moreover, the new domain-specific layers and the FC layers in the shared network are fine-tuned during the tracking process. The intuition behind this approach is that this learning manner helps obtain the parameters of the shared layers with the useful generic feature representations in order to track generic objects during the testing phase without pretrained domain-specific heads. By training the network in this fashion, the tracker becomes generic and not biased to any specific sequence. To increase the accuracy of the trackers, Danelljan et al. [57] introduced ATOM tracker (Accurate Tracking by Overlap Maximization), which consists of a deep neural net structure, which is inspired from IoU-Net [113]. The IoU-Net predicts the IoU scores of candidates in order to select the best estimate with the highest IoU value as the tracking result.

Inspired from the successful works in facial verification [198, 187], keypoint descriptor learning [235] and one-shot character recognition [125], Bertinetto et al. [23] introduced the Fully-Convolutional Siamese (FCsiamese) Networks for object tracking. According to the paper, the tracking problem is defined as constantly detecting the object frame per frame via a similarity function $f(z, x)$ that compares an example image z to a candidate image x of the same size. The similarity function is built on two full-convolutional networks, and each extracts relevant features for a robust similarity measure. The full-convolutional

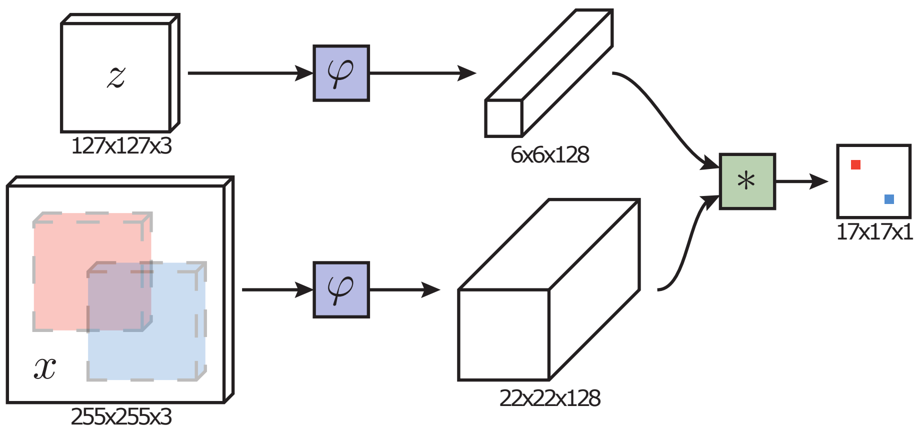


Figure 2.14 – Fully-convolution Siamese architecture. The color pixels indicating the high values of similarity map correspond to the sub-windows in the search area x [23]

Siamese architecture is shown in Figure 2.14. Given a search area x and an exemplar image z containing a target, those two images are mapped into an embedding via the function φ, those outputs are the inputs of a similarity function $f\left(\varphi(z), \varphi(x)\right)$ whose output is a scalar-valued score map. Concretely, the cross-correlation operation $f\left(\varphi(z), \varphi(x)\right) = \varphi(z) * \varphi(x) + 1$ is deployed to obtain the similarity measure and this make their model similar to the Correlation Filter scheme (Fig. 2.9) without update feedback. In terms of training data, the ImageNet Video dataset [183] (ILSVRC2015) containing almost 4500 videos with more than one million annotated frames was used to train their fully-convolutional networks.

Meanwhile, instead of using the cross-correlation as the similarity function, Tao et al. [203] train a matching function within a Siamese structure and treat the tracking problem as classification. Therefore, all the candidates sampled in the ROI at the current frame are classified into positive and negative groups. Not formulating the tracking problem as a classification problem, GOTURN (Generic Object Tracking Using Regression Networks) tracker by Held et al. [100] used a regression learning model to predict bounding boxes. Within the similar Siamese network architecture, the block of FC layers at the head of their network structure, as a regression network, is fed by the extracted features of both search area (in the current frame) and target object's template (from the previous frame) from two CNNs. The output of this regression network is the object's bounding box, which is then refined with a provided motion model to adapt to the empirical smooth movements. Another relevant variation of Siamese network was introduced by Valmadre et al. [208]. The method uses an asymmetric Siamese network that combined the conventional Correlation Filter structure with a Siamese network. Precisely, a CNN is pretrained to extract robust features from image frames as a pre-processing step before correlation. Another significant contribution of this paper is their formulation of evaluation and back-propagation of the Correlation Filter Network. Benefiting from the computation of correlation in the Fourier domain, this method outperformed other similar trackers in terms of speed. Recently, there have been many other trackers based on the Siamese structure such as as [219, 137, 249, 139, 218] and those trackers all achieved excellent results in terms of speed and accuracy. Inspired by the Region Proposal Network (RPN) of Faster RCNN [176] as the regressor of bounding boxes and the classifier of objectness, the SiamRPN tracker by Li et al. [137] includes a Siamese network for pre-processing step and an RPN for classification and regression. Indeed, the Siamese subnetwork serves to extract features from template and image frames. Next, the extracted features go through a Region Proposal Network which comprises one classification branch and one regression branch. Inside each branch, both template and image features go through a CNN, which aims to build k anchors for better prediction [176], before the correlation operation between these two outputs, i.e., Siamese-like structure. Then the correlation results in k pairs of layers are the k positive-negative sample pairs building a $2k$-layer response map. In the same manner, the regression branch takes the extracted features of template and image as inputs of the Siamese-like structure, but the CNN of this branch, instead, generates $4k$ features corresponding to 4 coordinates used for the proposal refinement of k anchors. The detail of SiamRPN is illustrated in Fig. 2.15. By adapting the Region Proposal Network for object tracking, the method technically resolved the pose and size-change issues during tracking. Since multiple similar objects may appear in the usual case of SOT videos, SOT trackers can be distracted from their primary target. Hence, in the paper of Li et al. [139], a gradient-guided network was introduced, called GradNet, which uses the gradients extracted from the Siamese network to update the target's template so as to differentiate it from other similar objects.

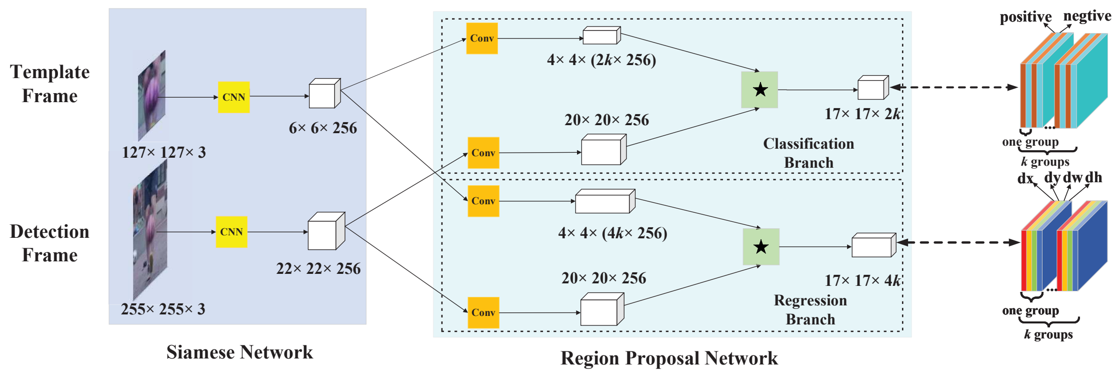


Figure 2.15 – Main framework of Siamese-RPN. ($\star$ denotes correlation operator) [137]

To conclude, according to the search strategy, most Deep tracking methods are mainly separated into two approaches: regression problem such as bounding box regression [100, 219] or correlation filter (Fig.2.14) [23, 208, 249] and classification problem (Fig.2.13) [162] or both [57, 203, 137]. In terms of network structure, they are divided into two groups: Siamese trackers [23, 208, 249, 219, 203, 137, 249, 139, 218] and non-siamese trackers [100, 162, 57].

2.4 Single view multi-object tracking

Despite the tremendous advances of single object tracking algorithms, Multiple Object Tracking (MOT) by applying multiple Single Object Trackers is still not practical in many applications, which require real-time processing, such as tracking in autonomous driving. This is because the single object tracking problem ignores many critical issues of the multiple objects tracking in reality. First, the computational cost is proportional to the number of objects appearing on the scene. Many single object trackers can perform real-time tracking (i.e., *>30 fps*) with high accuracy, but only track one single object in the video. Secondly, single object trackers focus on differentiating the object's appearance from the background, while in most MOT applications, the tracking algorithms need to track many objects which belong to the same category such as people, vehicles. The SOT tracker might confuse its target with similar objects since the SOT trackers do not differentiate intra-class objects. This leads to poor tracking results due to the segmentation of the trajectories of multiple objects. Third, MOT algorithms need a detector to detect the potential target *every frame* and initialize new trackers from the new detections while excluding false positives. Next, the SOT trackers perform under the assumption that the target object is always present in the entire video; however, in MOT videos, an object might be absent temporally. Finally, since a target can appear and then disappear multiple times for many reasons such as occlusion or out of Field of View (FOV), the re-identification is required to reconnect the target's trajectories. These difficulties keep SOT algorithms from being adaptable to real-world demands. In the following sections, we will discuss MOT state-of-the-art methods.

In the literature on tracking, there are two main tracking communities working on different contexts: the *visual tracking* community, which uses video streams as input, meanwhile the *multi-sensing tracking* community which mainly uses multiple sensors to detect and record target motions by radio signals. Visual tracking is widely known in computer vision with various applications based on images recorded by cameras. Otherwise, multi-sensing tracking is popular in the robotic and control field with many applications from civil to military. In the next subsection, we introduce the most relevant approaches in multi-sensing tracking that have great impacts on the visual Multi-Object Tracking algorithms. In this community, the term *Multi-Target Tracking* (MTT) is mainly used instead of *Multi-Object Tracking* (MOT). For convenience, in this dissertation, the term *Multi-Object Tracking* (MOT) is used in all cases.

2.4.1 Classic methods

Since the success of tracking techniques in remote sensing, which relies on sensors to acquire signals, they have become popular in engineering and military fields, and many MOT trackers developed later have mainly relied on those techniques. The most popular one is the Kalman filter and its variations such as extended Kalman filter (EKF), unscented Kalman filter (UKF), unscented extended Kalman filter (UEKF) [31]. In general, the Kalman filter (known as Linear Quadratic Estimation) models a dynamic system which is presented through the state of a discrete-time process that is expressed by a linear stochastic equation system. The state of the process at the time instance t, x_t, equals to a state-transition model F multiplying by the previous state at $t-1$, x_{t-1}, with some process noise w_t, as the formulations:

$$x_t = Fx_{t-1} + w_t \tag{2.1}$$

In tracking problem, the state is chosen as the position of the target, the transition model predicting the future target's position is based on its *trajectory*, and an observation model obtains the position measure from *image input*. However, to model an arbitrary motion of objects, the linear system does not fit in general cases. Therefore, the non-linear versions of the Kalman filter, such as extended Kalman filter (EKF), are used instead. Concerning the extended Kalman filter, the functions of the state transition and observation model are not necessarily linear but should be differentiable. The EKF adapts the multivariate Taylor Series expansions to linearize a non-linear model at the working point. To sum up, there are two steps in the EKF: first, predicting state and covariance estimates; secondly, updating the filter from the measurements (or the *observations*). In 1979, Reid et al. [175] introduced the first multi-target tracker based on the Kalman filter, called Multiple hypotheses tracker (MHT). The data association strategy used in the MHT is to delay giving final results after a certain time. The delay allows the algorithm to open up all the hypotheses, which are all the combinations of detections over the period of the k-last frames, called the *hypothetical tracklets*. In other words, considering only the k-last frames causes the elimination of the hypotheses older than k frames. The MHT possesses multiple track trees at each time instance, and each tree represents all the hypotheses with a root growing from a single observation. At each new frame, the track trees are updated from new observations (i.e., new detections), and each track (i.e., *branch*) in the trees is given a score which is based on how likely the track can be formed from its detections. The best set of non-conflicting tracks (called *the best global hypothesis* in the original paper [175]) can then be found by solving a Maximum Weight Independent Set problem. Therefore, the branches, which are way far from the roots, are pruned off the trees. Subsequently, the MHT has the capability to explore solution space intensively, but its setback is the unnecessarily increasing number of targets which are mostly false positive. That causes the difficulties of implementing the MHT in practice due to the high computational cost. There were several works trying to solve this typical problem such as propagating only the M-best hypotheses [52] or extending the classical particle fitler [108]. The MHT tents to spawn new tracking segments that might belong to the same object (called *tracklets*). In many cases, failing to detect object eventually leads to terminating good in-process tracklets and creating new ones. Hence, the MHT is unable to recover the failure tracking cases. In 2015, Kim et al. [123] revisited the MHT in the case of visual tracking with a new appearance model integrating in the conventional target scoring function. The promising results obtained are comparable to the state-of-the-art performance.

As one of the most popular tracking moving objects methods in the early 80s, Joint Probabilistic Data Association (JPDA) filter [82] was widely used in the robotic community. It is an elegant method of associating new detections with existing targets using a joint probabilistic score. One of the advantages of this approach compared to the MHT is that it allows objects to be assigned with dummy nodes, which represent missing detections. In 2015, Hamid et al. [93] revisited JPDA filter and applied the method in visual tracking. By reformulating the data association problem as a bipartite graph problem and solving it via Interger Linear Programming (ILP), they succeeded in adapting JPDA approach to the visual tracking problem. One of the biggest problems while implementing JPDA tracking method in practice is that finding the exact solution of JPDA is NP-hard. To tackle this issue, Oh et al. [163] presented the Markov chain Monte Carlo data association (MCMCDA) for solving data association problems arising in multiple-target tracking. The authors have proved that, in order to track a single target, the single-scan MCMCDA algorithm provides a fully polynomial randomized approximation scheme for JPDA. They also proposed the multi-scan MCMCDA algorithm to track an unknown number of targets.

Using particle filtering as [108], the MCMC-Based Particle Filtering (MCMC-PF) method by Khan et al. [122] addresses the problem of managing the entries and exits of multiple targets, which probably have the same appearance and frequently interact each other. In detail, the method uses a Markov Random Field (MRF) motion prior to tackling the identity-switching issue that occurs when multiple objects overlap. The authors then introduced the Markov chain Monte Carlo (MCMC) sampling approach to reduce complexity caused by the MRF formulation.

Vo et al. [213] have developed another well-known tracking technique, which named Gaussian Mixture Probability Hypothesis Density Filter (GMPHD), to track multiple moving objects. However, this method is dedicated to detecting moving objects in noisy detection environments. This filter is implemented in multi-sensing tracking rather than in visual tracking.

2.4.2 Tracker management for Single-Object-tracking based approaches

Processing independently from the multi-sensing tracking community, the visual tracking community early focused on tracking a single object. Notwithstanding, as aforementioned, applying multiple single-object trackers to track multiple objects simultaneously and parallelly is being challenged by many critical issues. One of the most significant issues is how to keep the parallel tracking process operating while objects can appear and disappear at any time due to a variety of reasons such as entering/leaving the scene, occlusion by obstacles or by other objects, i.e., *mutual occlusion*. The mutual occlusion might be the biggest problem in MOT because it causes trackers to drift, switch their identities, or "stick" together. These issues demand the Single-Object-Tracking based (SOT-based) approaches the ability to organize all trackers and keeping trackers active or inactive in appropriate situations, e.g., targets appear and disappear temporarily.

There are many works [229, 47, 184, 252, 231, 21, 232] addressing this typical issue. In 2015, Xiang et al. [229] specifically addressed this issue and emphasized the importance of *lifetime management* while implementing Single Object Tracker to track multiple targets. The authors proposed an online MOT framework which supports the SOT implementation on the MOT context by introducing their Markov Decision Process (MDP) formulation in modeling the lifetime of a single object. The next section is devoted to describing this work.

2.4.2.1 Markov Decision Process in tracker management

A Markov Decision Process (MDP) is a discrete-time stochastic process that models a decision-making process whose evolution over time is under a decision-maker with some probabilities at each time step. A MDP is defined through a tuple of objects $(\mathcal{S}, \mathcal{A}, P_a, R)$:

- The target state space $\mathcal{S}$

- Sets $\mathcal{A}(s)$ of available actions at state $s \in \mathcal{S}$
- $P_a(s, s') := P(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$
- A transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$
- A reward function $R(s, a)$ is the immediate reward received after taking action a from state s.

The lifetime of a single target is modeled by a deterministic Markov Decision Process (MDP), which means $P_a(s, s') \in \{0, 1\}$. The state-space mainly consists of 4 states: *Active*: corresponding to the initialization of a new tracker from a new detection, *Inactive*: the ending of any tracker; *Tracked*: the in-tracking-process of trackers; *Lost*: the temporally losing target of trackers. The MDP framework for Multi-Object Tracking is illustrated in Figure 2.16. In their framework, all new detections, which do not belong to any current tracking target, are first set to *Active* state, then the true positive detections moves on to *tracked* state, otherwise the *false positive* detections are eliminated by setting them to *inactive* state. The trackers in *tracked* state continue tracking their target if there are no difficulties such as occlusion, or else, they are paused by transiting to *lost* state. The *lost* trackers restore their *tracked* state if there is any true corresponding detection found in their nearby area where they lose their targets. Otherwise, they retain their *lost* state in the next frame. Additionally, all *inactive* trackers are stored as tracking results and never come back to the tracking process.

In terms of actions and transitions, there are several possible transitions designed which allow trackers to be transferred between target's states. In the framework of Xiang et al. [229], all the actions are deterministic, i.e., the next state of a tracker is determined by its state and the taken action at the current frame. In detail, at *active* state, a binary Support Vector Machine (SVM) is implemented to classify all detections into false positive and true positive groups. All true positives take the action a_1 in order to pass to *tracked* state, otherwise, the action a_2 move all the false positives to *inactive* state. Each *tracked* tracker performs a SOT process independently if it succeeds to track its target, it takes the action a_3 and remains in *tracked* state, in the case of failure, a data association step follows up to link the tracker with the target via nearby detections. Indeed, the success of the data association (DA) step is decided by a second SVM. If DA step succeeds, the tracker updates the new appearance of target and remains *tracked* state, if not, it is set

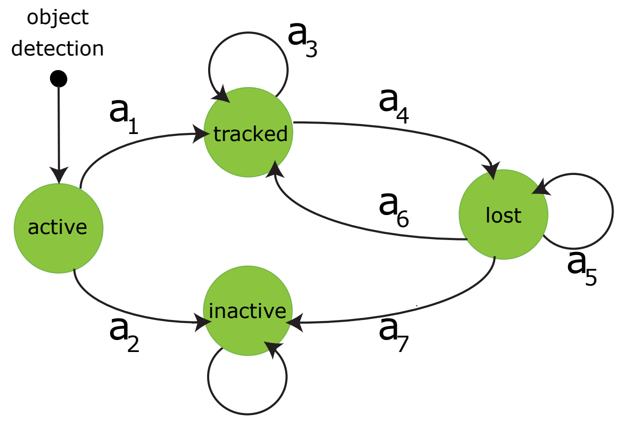


Figure 2.16 – The Markov process of a single object in Multiple Object Tracking Xiang's framework [229]

into *lost* state by taking the action a_4. In reference to *lost* trackers, a matching process tries to recatch the target from all nearby detections via data association. In the case of success, the action a_6 takes the trackers back to *tracked* state in the next frame; otherwise, they keeps their *lost* state with the action a_5. Moreover, after a fixed time T, every tracker still in *lost* state is transferred to *inactive* state by action a_7. The reward function might be used to determine the deterministic policies during training [229], which is described in *Reinforcement learning problem* (RL). The reinforcement learning problem is used to determine the best policy for MDP to accumulate the maximum reward at the end of the process. In the paper [229], the authors present their method as an Inverse RL algorithm. The authors claimed the use of *Inverse Reinforcement Learning* (IRL) algorithm to design the reward function. However, the goal of the IRL problem is to determine the reward function with a given specific policy, which is unknown in this case. Another issue spotted in the paper is that the policy applied at *tracked* state is the outcome of a SOT tracker (or the data association step), which should be a function of the appearance of the target. This means that their transition function $T(s, a)$ does not map the state s (e.g. *active, inactive, lost, tracked*) and action a. Their MDP is technically not a standard MDP and does not solve the initial IRL problem. Instead, the authors use the regular condition loops to guide their MDP in the tracking process.

Besides the introduction of MDP to naturally handle appearance/disappearance of targets, to well organize their trackers, they also developed a mechanism to treat all trackers in order. Concretely, given a new input frame, they first run the *tracked*-state tracker, this favors these trackers to gather first the detections belonging to them via back-up DA step. The remaining detections are used to recover the *lost*-state trackers. The similarity scores between the *lost*-state trackers and these detections are computed in order to assign the *lost*-state trackers with the detections via the Hungarian algorithm [161]. The rest of the detections serves to initialize new trackers.

2.4.2.2 Spatial-Temporal attention to handle multual occlusions

As one setback of Xiang's MDP framework [229], the framework uses a simple technique to detect occlusions, which is a significant issue that deteriorates the tracking results. Indeed, their method relies on forward-backward Lucas-Kanade keypoints matching, which is later scored by an SVM. To resolve this problem, many papers [47, 184, 252, 231, 21] focuses on spatial-temporal relation between targets in order to predict occlusion efficiently, when targets start crossing each other. Chu et al. [47] introduced a novel approach that can handle normal or mutual occlusions using a spatial-attention network to retrieve features on the useful zone of attention inside the bounding boxes of targets, called *visibility map*, during occlusion. Meanwhile, Zhu et al. [252] proposed a novel MOT framework supporting SOT-tracker. Their framework performs in the same manner as those of Xiang et al. [229]. Figure 2.17 shows the pipeline of the MOT framework, which consists of 3 main stages: detection, single-object tracking, and data association. Two points make this method different from the MDP of Xiang et al. [229]. First, they introduced a novel network architecture, named Dual Matching Attention Networks (DMANs) including the Spatial Attention Networks (SANs), which aims to compare detections with tracklets to extract useful spatial features on the detections while avoiding the occlusion part, and a Temporal

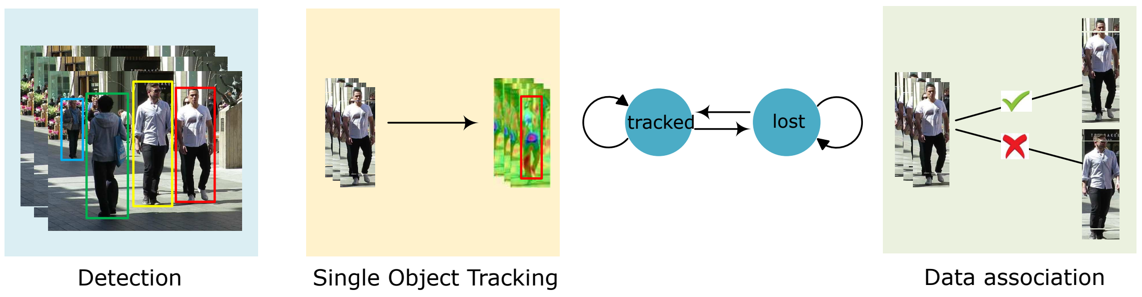


Figure 2.17 – The MOT pipeline of [252]. The state of each target switches between tracked and lost depending on the tracking reliability. Single object tracking is applied to generate the tracklets for the tracked targets while data association compares the tracklets with candidate detections to make assignments for the lost targets[252].

Attention Network (TAN) which uses the extracted features as inputs to associate tracklets with detections. Secondly, their approach matches each tracklet with a detection in the detection set, which is technically a *bipartite graph* matching in their data association step. Recently, Berhmann et al. [21] presents an MOT framework, called Tracktor, to handle the track-occlusion-state switching of targets. This method uses two neural nets with a Faster RCNN backbone to build one classifier for occlusion detection, and another for the regression between targets' predictions and detections.

2.4.3 Tracking-by-Detection paradigm

As most of the surveillance applications focus on human tracking, in particular, the last decade marked the return of the Tracking-by-Detection strategy [114, 237, 4] due to the emergence of efficient person detectors [4]. By definition, Tracking-by-Detection is to gather the detections belonging to the same identities in a whole video to infer their trajectories. This approach possesses several advantages ahead of those of SOT-based. First, not having any tracker, the Tracking-by-Detection algorithm only focuses on finding the repeat of individuals from the collection of all detections and matching those detections with different identities. Therefore, they do not have to cope with the typical problems of SOT-based methods mentioned previously. Because no trackers are used in the Tracking-by-Detection approaches, the tracking algorithms do not need to organize the order of trackers, to manage the active and inactive state of trackers, to update the target appearance without causing drift problems, to have a robust appearance model resisting to complex scenes or the environmental lighting changes. Concerning detectors, the modern detectors can work in any condition of environments unless targets are unrecognizable. Notwithstanding, the downside of this approach is that it is sensitive to the reliability of the detectors. This causes the Tracking-by-Detection approach to deal with several specific problems. First of all, in most cases, false positives appear frequently due to complex scenes, environment textures, reflections, or deformed images caused by cameras and shadows. Secondly, multiple detections can overlap on the same person, and this leads to the need of a post-processing step to induce the final tracking results. Thirdly, when detectors fail to detect targets, i.e., *true negatives*, this makes their trajectories broken into small segments (called *tracklets* as well). Hence, these tracking methods must be able to estimate hypothetical positions of targets in the case of missing detections. Lastly, because of resorting detections regarding each individual, data association approaches are in favor of post-processing after gathering all detections. Those approaches seem unsuitable for many applications requiring online tracking results, but in the literature on data association approaches in the next subsection, there is a significant number of papers reformulating the problem to adapt to the online processing requirement.

2.4.4 Data association

Since the management of the different targets is the main challenge for MOT, the Tracking-by-Detection paradigm has evolved as the main approach. This is especially true since the advent of high-performing category detectors. The main objective of Tracking-by-Detection methods is to match targets, i.e., objects identified at frame t, to detections, i.e., resulted bounding boxes from detection at frame $t+1$, which is formulated as a *data association* problem. In this section, we review the most common data association methods used in the MOT tracking problem.

2.4.4.1 Minimum Cost Bipartite Matching

Matching the current targets with the new detections found in the next frame is formulated as a bipartite matching problem. Indeed, given two independent sets, one contains all targets at the current frame, another one contains all detections found in the next frame image, we want to build a *bipartite graph* (or *bigraph*) $G = (T, D, E)$ in which T and D are respectively the target and detection sets, and each edge $e \in E$ represents an assignment linking a target to one or many detections. Meanwhile, a single detection $d \in D$ can not be matched with more than one target $t \in T$. Additionally, each edge has a non-negative cost $c(t, d)$, which is an appearance distance between targets and detections. We want to find the best matching between *two consecutive frames* with a minimum total cost. A variation of the Hungarian algorithm by Kuhn [128] can solve the minimum cost bipartite matching problem efficiently with the complexity $\mathcal{O}(n^2m)$ where n and m are the numbers of targets and detections respectively. In order to handle the entries, exits, or missings of targets, some papers introduced an extended bipartite matching problem by adding *dummy* or *virtual* nodes. According to these methods, m targets link at the current frame t links n detections found at the next frame. n virtual *entry* nodes are added to the target set T to stimulate the creation of new targets in the next frame. Similarly, m virtual *exit* nodes are also added to the detection set D to prepare for the case of all targets exit in the next frame. Due to the matching occurs only between the current frame and the last one, these methods fit for online tracking applications. Figure 2.18 depicts the extended bipartite matching problem in data association tracking approaches. Recently, Xu et al. [232] proposed an alternative approach of Hungarian assignment to the bipartite matching problem, which uses two Gated Recurrent Unit (GRU) to encode targets' and detections' features for another deep neural network, so-called Deep Hungarian Net (DHN). Indeed, the output of the DHN is a "pseudo" assignment matrix, whose elements are bounded between 0 and 1. To conclude, the papers treating data association as Minimum Cost Bipartite Matching

(MCBM) problem include [4, 191, 184, 252, 224, 232].

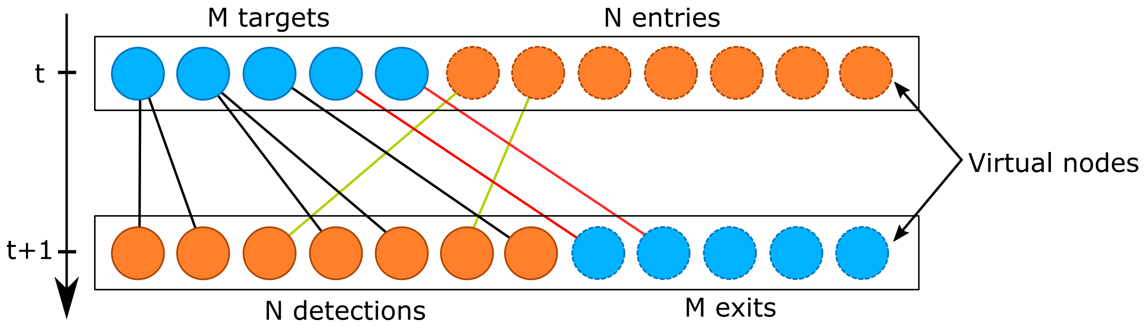


Figure 2.18 – The extended bipartite matching problem in data association tracking.

2.4.4.2 Maximum Weight Independent Set

The data association problem has also been reformulated as a Maximum Weight Independent Set (MWIS) problem. Given a set of detections $\mathcal{D}$ during a time period from t_1 to t_2, the trajectories of targets are the independent subset of detections (i.e., *tracklets*) such that any single node of the independent set is only found in a single time frame. Hence, the tracklets are all the possible combinations of detections under the previous conditions. These tracklets are designated as the nodes of a graph. Those nodes have the *weights*, which represent the likelihood of the corresponding tracklet among the other tracklets. The edges of the graph will connect the nodes (sets of detections) if they share *any same detections*. The illustration of this type of graph is shown in Figure 2.19.

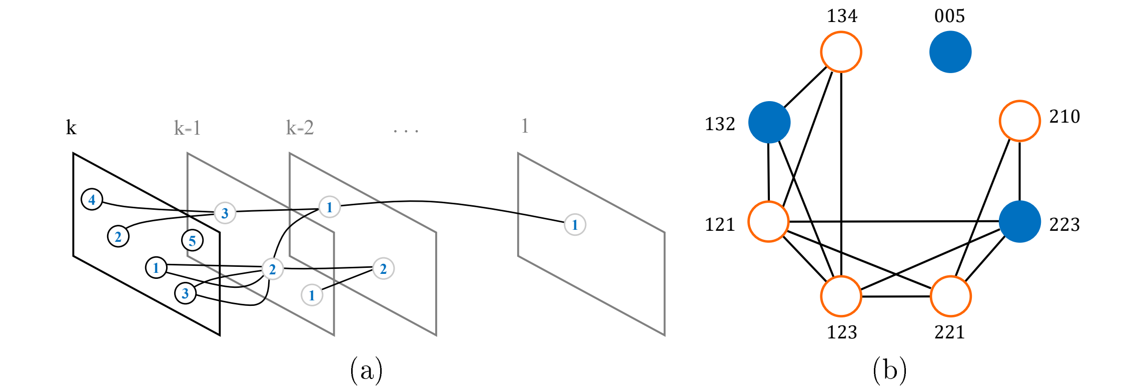


Figure 2.19 – The illustration of MHT. (a) Track hypotheses after the gating test at time. (b) An undirected graph with Maximum Weighted Independent Set (MWIS) is highlighted in blue[29].

Therefore, we can see the trajectories of the target as the Maximum Weight Independent Sets. However, the MWIS problem is known to be NP-hard, and it can be solved approximately, with the complexity of $\mathcal{O}(n^3)$ (n is the number of nodes in the graph). This motivated Brendel et al. [29] to introduce a variant of the MWIS problem devoting to matching detections in two consecutive frames, that reduces the complexity to $\mathcal{O}(n^2)$. Indeed, the authors redefined the node as a *pairs of consecutive detections* (2-frame tracklets); this means they only consider the time period from t to $t+1$. As a result, the trajectories are created from the tracklets consecutively sharing the same detections in the illustration 2.20 (a). Therefore 2-frame MWIS can be solved optimally and efficiently in

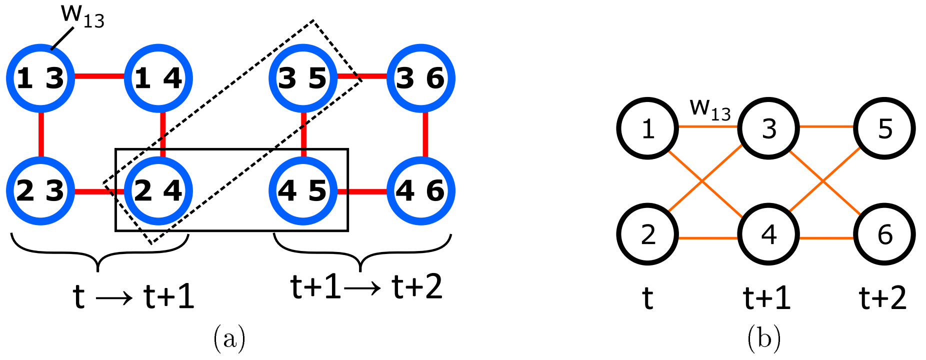


Figure 2.20 – The graph: (a) Blue nodes (tracklets) are connected by edges at the time frame from t to $t+1$, and share the same detection (denoted with integers); this partitions the graph into two independent subgraphs[29]. (b) The representation of MWIS in bipartite matching graph problem (Ristani thesis)

$\mathcal{O}(n^2m)$ time for n detections and m tracklets. With a time frame of 2, the MWIS problem can be converted to Minimum Cost Bipartite Matching by redefining the weight of edges (see Fig. 2.20 (b)) and also the semantic of nodes. In the Multiple Hypotheses Tracking (MHT) revisited paper, Kim et al. [123] also resolved the MWIS problem to find the best hypotheses in the last N frames, i.e., the best tracklets of N-frame. Otherwise, Choi [45] tracker (Near-Online Multiple Target Tracking NOMT) also uses the MHT at his final step to release the target's trajectories. Figure 2.19 describes the MWIS problem in the MHT method.

2.4.4.3 Minimum Cost Network Flow

Zhang et al. [237], Pirsiavash et al. [167], Zhang et al. [240] reformulate the data association in MOT as a Minimum Cost Network Flow (MCNF) problem. Given a set of detections, a directed graph is built from these detections as its nodes, and the edges represent the association of bounding boxes in time.

Additionally, there are two more especial nodes: *source* and *sink*. The intuition of this approach is that the creation of "streams" from *source* node to *sink* node correspond to trajectories of targets from its starting moment to its ending. The *source* node connects to any node in the network, except for the *sink* node, in order to initiate new tracking processes of all newly detected targets at any frame of the video. Otherwise, apart from the *source*, all nodes connect to the *sink*; thus, the *sink* is used to terminate the tracking process of any target at any moment. The conventional nodes are connected by the edges weighted by an appearance distance between their representing detections in 2 consecutive frames. The trajectory of a target is modeled by a path which starts from the *source*, then goes through the detection nodes which are linked by minimal cost edges and ends at the *sink*. The formation of tracklets in the network looks similar to the phenomena of a stream from one point to another, its path is located at low energy places. An illustration is given in Fig. 2.21.

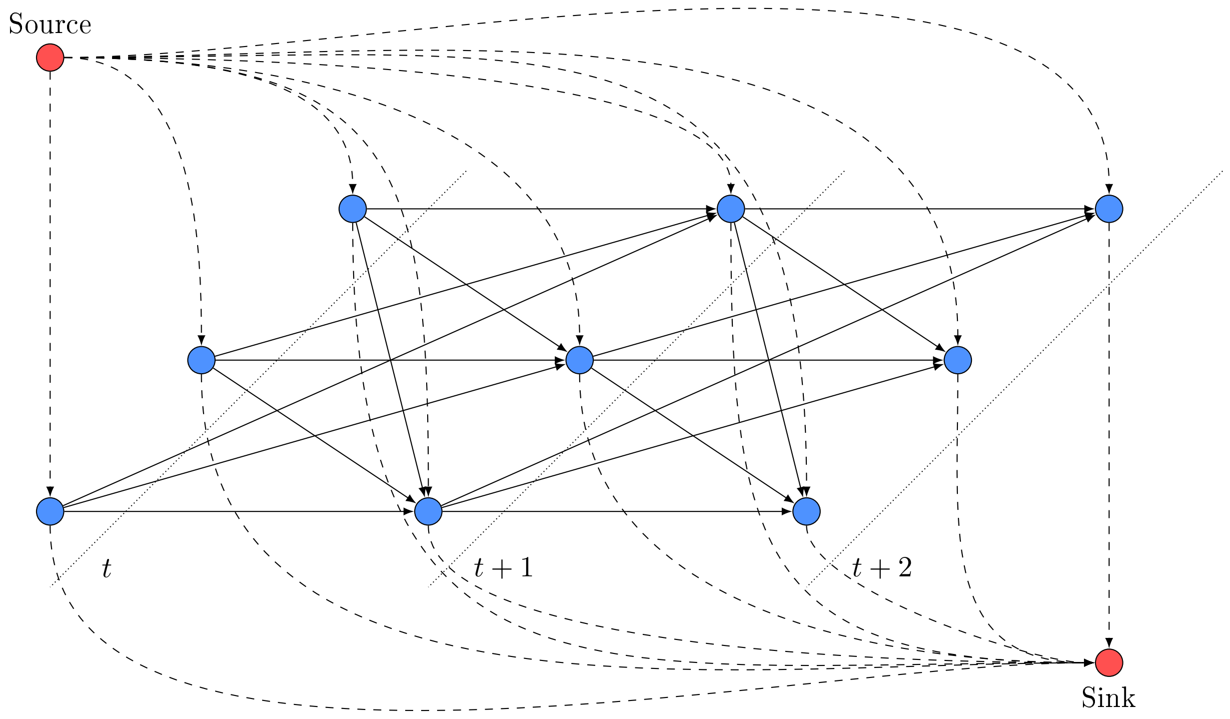


Figure 2.21 – The network flow in Multiple Object Tracking

There are several approaches to handle the missing detections (i.e., *true negatives*), long-time, and mutual occlusions. The authors [240] introduced virtual nodes that could link to the previous detection nodes with a certain cost, which avoids the flow network abusing this type of nodes to create "endless" trajectories. In the paper [237], the occlusion problem is modeled as merging splitting paths. Moreover, the authors [237] also added in their network flow model an additional node just after every regular node, they are connected by an *observation edge* with a cost which is opposite to the confidence score of detections. This takes into account the false positives generated by detectors. The algorithm of [237] has a complexity of $\mathcal{O}(n^2 m \log n)$ for a graph with n nodes and m edges. Pirsiavash et al. [167] state that if n and m scale linearly with the number of frames N, then the algorithm of Zhang et al. [237] runs in $\mathcal{O}(N^3 \log N)$ to find K tracks. Noting that the graph has unit capacity edges and is directed and acyclic, Pirsiavash et al. [167]

provided an algorithm that solves the network flow problem in $\mathcal{O}(KN \log N)$, involving the Successive Shortest Paths and the bisection search over K. They also give a dynamic programming greedy approximation algorithm with $\mathcal{O}(KN)$ complexity. This algorithm proves as effective as the one involving the Successive Shortest Paths. The Minimum Cost Network Flow (MCNF) graph in the papers [237, 167, 240] is the generalization of the standard MCBM formulations as it covers the effects of false detections, the starting, ending or temporally missing of targets.

In 2015, a variation of Minimum Cost Network Flow (MCNF), named *Pairwise Costs Network Flow* (PCNF), was introduced by Chari et al. [37]. They use the same graph as in the Minimum Cost Network Flow problem but extend the optimization objective to account for all pairwise costs. The extension aims to resort to the multiple detections per target, which usually happens when using detectors. Previously, in the most common way, especially in an undirected graph formulation, a Non Maxima Suppression (NMS) step [237, 200, 201, 4, 240] usually follows after gathering detections from detectors. A problem emerging when detecting targets in a crowded scene is that the mutual occlusion results in the detection overlapping. Applying the NMS also excludes many reliable detections. Addressing this problem, *Pairwise Costs Network Flow* (PCNF) builds the internal edges between detections within a frame, which are not allowed in a standard network flow. These edges are weighted by an overlapping score, e.g., IoU score. As a result, the cost of a path (trajectory) includes all pairwise costs in the trajectory, rather than only costs of edges along the trajectory. The authors resolve the problem via Integer Linear Programming.

2.4.4.4 Graph Multicuts

There were many noticeable papers reformulating data association problem for Multiple Object Tracking as a Graph Multicut Problem (MP) [200, 201, 119]. Tang et al. [200] constructed an undirected graph $G = (V, E)$ whose nodes V represent all detections in the whole video and whose edges connect pairs of detections belonging to the same target, including those within the same frame. The solution of the Minimum Cost Subgraph Multicut Problem is the subgraphs $G' = (V', E')$ of G, which only includes the detections from the same targets. Every node and edge are assigned respectively to a *unary* and a *pairwise* cost. The unary cost is usually assigned the confidence score of detections to account for the false-positive effect of detectors, and the pairwise cost is usually the affinity between two detections. The graph multicut problem can be solved efficiently using the duality between maximum subgraph weight [200, 119, 179] and minimum total cost of cut [201, 202]. In the paper [200], the graph multicut problem is interpreted as the minimization of the cost to retain nodes and edges in the subgraphs, which indicate the distinct targets. This means that each subgraph containing an unique target should have a negative cost. The minimization problem is then solved via the Binary Integer Program (BIP). The graph and subgraph solutions are depicted in Figure 2.22. Graph multicut was preliminarily introduced in the paper of Ristani and Tomasi [179] in 2014 with their formulation under the terms *correlation clustering* or *graph partitioning.* In fact, the correlation clustering problem can be obtained from graph multicut by removing the unary cost and inversing the pairwise cost to positive, which makes the original minimization problem in Multicut problem papers become the maximization problem in the formulation

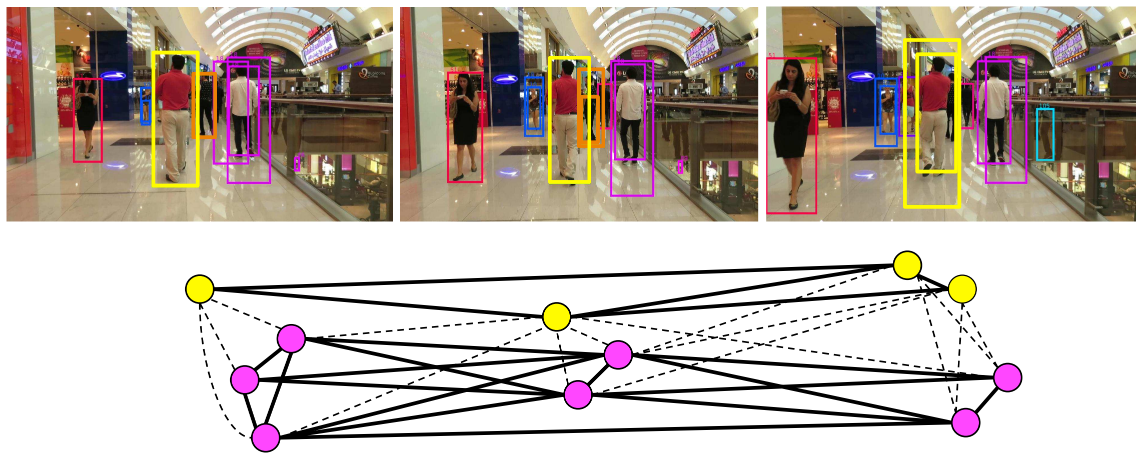


Figure 2.22 – An example for tracking by multicut. A graph (bottom) is built based on the detections in three frames (top). The connected components that are obtainedby solving the multicut problem indicate the number of tracks (there are two tracks, depicted in yellow and magenta respectively) as well as the membership of every detection[201].

of [179]. Graph multicuts or correlation clustering is an NP-hard problem [11], Tang et al. [200] proposed a heuristic solution for the unconstrained set partition problem by using Kernighan Lin algorithm [118]. Meanwhile, Ristani and Tomasi [179] rely on the optimization algorithms for graph partitioning in the literature [8]. Seeing the difficulties dealing with long-term occlusion or missing detections that leads to the segmentation of trajectories in tracking results, Tang et al. [202] introduce the Lifted Multicut Problem (LMP). It originates from the standard multicut problem [201] when redefining the regular edges and adding the *lifted* ones. In LMP method, the conventional edges connect the detections not further than T frames apart, which results in short paths; otherwise, the other detections can be connected by the *lifted* edges, which associate the short paths to form complete trajectories. Figure 2.23 illustrates the advantages of LMP against the conventional MP. Lastly, the optimization in LMP is solved via APX-hard [66].

2.4.4.5 Generalized Minimum Clique

Another formulation of data association in Multi-Object Tracking is the Generalized Minimum Clique Problem (GMCP), which was introduced by Zamir et al. [236]. Similarly to the graph formulation described in graph multicut previously, an undirected graph is constructed whose nodes represent detections and edges represent their affinity. Based on this graph, the MOT data association is reformulated as finding the nodes separated from disjoint clusters indicating distinct targets. In the main graph, these nodes form themself a complete graph, called a *clique*. Each clique contains all detections of one specific target and has a cost, which is defined as the sum of the appearance distances between each pair of the target's detections. The objective now becomes to find the cliques whose cost is minimum. The authors proposed an iterative algorithm to associate detections into identities iteratively, one clique (identity) per iteration. Every time a minimum clique is computed, its edges and nodes are removed from the graph, and the process is repeated

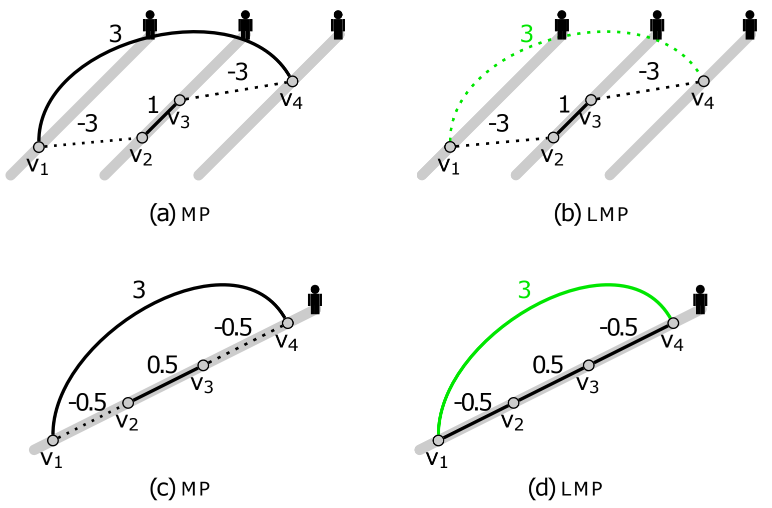


Figure 2.23 – Ground truth trajectories are shown in grey, regular edges in black, and lifted edges in green. In the optimal solution solid lines indicate co-identity, and dashed lines indicate cuts. Correlations are shown on each edge. (a) An example where MP incorrectly merges v1 and v3 whereas (b) LMP does not have evidence of regular path connectivity for this long range association. (c) MP incorrectly fragments the true trajectory whereas (d) LMP makes a correct assignment due to the lifted edge[202].

until the graph is empty. An illustration of one iteration is shown in Figure 2.24, where one detection from each frame is selected to produce the minimum weight clique. In practice, since finding the entire graph containing all detections of the whole video is extremely expensive, the GMCP tracker divides the video into multiple short sequences to limit the number of detections, then solving the GMCP results in the tracklets of all individuals. Therefore, after having all tracklets in the videos, they solve another GMCP again, but this time instead of using detections as nodes of the graph, they use the obtained tracklets form the last GMCP to determine the complete trajectories of targets in the entire video. Another variation of GMCP is presented in the paper of Tesfaye et al. [204] as the *Constrained Dominant Sets* problem. The difference is that they reformulated the graph partitioning problem in order to be able to solve it via the quadratic program.

Another sort of graph clique problem for data association MOT is introduced as Generalized Maximum Multi-Clique Problem (GMMCP) by Dehghan et al. [64]. In a similar way, they build a k-partite complete graph whose edges pair every couple of nodes (detections), which are not in the same cluster (set of detections in the same frame). In the case $k = 2$, the graph takes the bipartite form, so the problem becomes Minimum Cost Bipartite Matching problem. Using the same strategy in [236], the method solves the GMMCP twice to have the complete target trajectories. In the first time, the input of the GMMCP algorithm is low-size tracklets (maximum 10 frames long), the algorithm output the mid-size tracklets. The output is the input of the GMMCP algorithm for the second times to obtain the complete trajectories. In addition to the two-stage GMMCP pipeline, the method expresses the maximization problem in terms of Binary Interger Program (BIP); these make it capable of solving the assignment problem jointly and optimally for all iden-

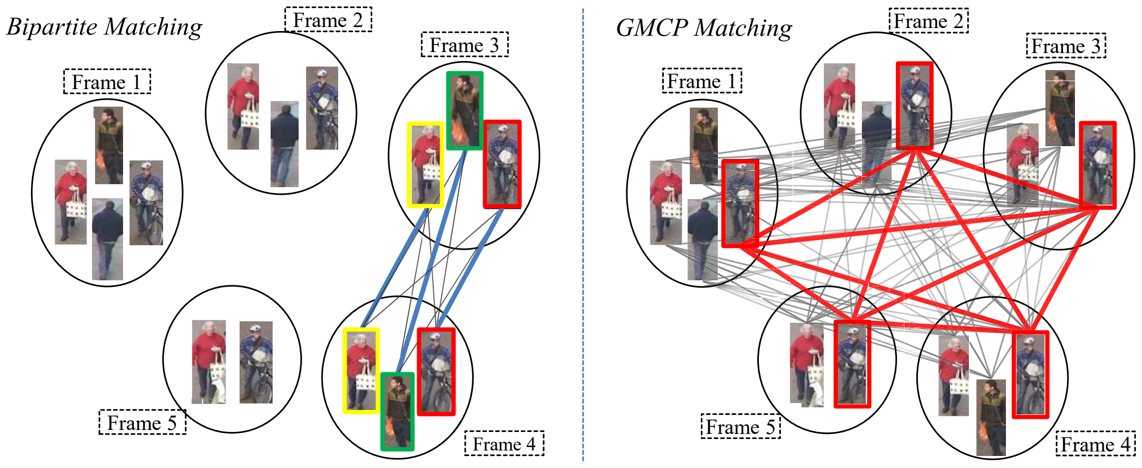


Figure 2.24 – Bi-partite vs. GMCP matching. Gray and colored edges represent the input graph and optimized subgraph, respectively. Bi-partite matches all objects in a limited temporal window. On the other hand, the proposed method matches one object at a time across full temporal span, while incorporating the rest of the objects implicitly[236].

tities in a reasonable time, even though, GMMCP has been proved to be NP-hard [64]. An illustration of the approach is shown in Figure 2.25.

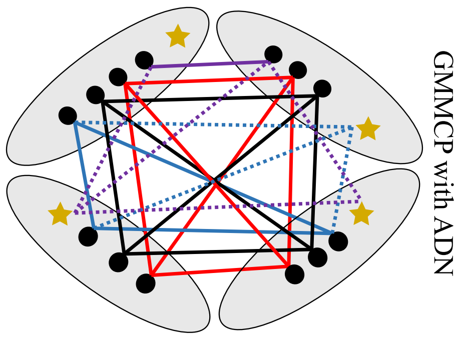


Figure 2.25 – An illustration of the Maximum Weight Multi-Clique problem. In the graph, 4 cliques are found, each shown in a different color. The method uses virtual nodes or Aggregated Dummy Nodes (ADN) represented by stars to efficiently model occlusion phenomena[64].

2.5 Multi-view object tracking

In this section, we focus on the Multiple Target/Object Multiple Camera tracking (MTMC) problems. In the literature, there are three main problems that can be solved by a camera network: using overlapping camera networks improving the robustness of tracking algorithms, e.g., handling occlusions, preventing identity lost/switches; non/overlapping camera networks tracking targets in a large area; re-identification or person search. The diagram 2.26 shows the main categories of multiple camera systems.

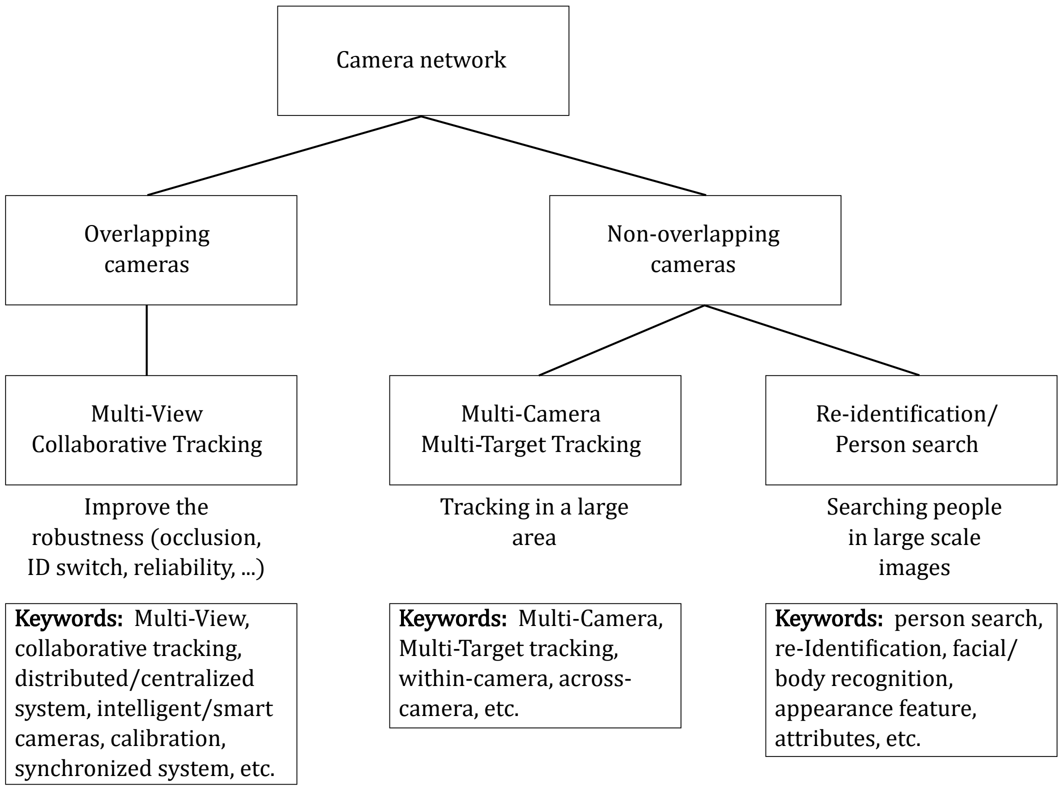

Camera network
Overlapping cameras
Non-overlapping cameras
Multi-View Collaborative Tracking
Multi-Camera Multi-Target Tracking
Re-identification/ Person search
Improve the robustness (occlusion, ID switch, reliability, ...)
Tracking in a large area
Searching people in large scale images
Keywords: Multi-View, collaborative tracking, distributed/centralized system, intelligent/smart cameras, calibration, synchronized system, etc.
Keywords: Multi-Camera, Multi-Target tracking, within-camera, across-camera, etc.
Keywords: person search, re-Identification, facial/ body recognition, appearance feature, attributes, etc.

Figure 2.26 – Overview of Multiple Camera Multiple Target/Object tracking problems. Keyword boxes list the commonly used terms for each tracking/re-ID problem.

2.5.1 Architecture of camera networks

In practice, the camera systems using different MTMC tracking algorithms possess different architectures. According to the computational model, we classify them into two main models: Centralized Multi-Camera Network and Distributed Multi-Camera Network. In centralized computing, all cameras connect to a computation center; the video streams or any detection information is transferred from each camera to the computation center. With this architecture, each camera usually transfers the detection information including bounding boxes, the extracted features, etc. or even the video stream sometimes. This model is used to collect data, detections for offline processing such as offline tracking, re-identification, or people searching. On the contrary, distributed computing distributes the computational charge to all cameras inside the network, i.e., *Smart Cameras*. All cameras are directly or indirectly connected together. Performing multi-cameras tracking, in this case, means that each camera performs its own tracking task, all cameras exchange their tracking results, and collaborate with each other by this information when they need to track multiple targets more efficiently such as handling mutli-entry/exit, occlusion or reconnecting trajectories of the same targets across cameras. The information exchanged usually consists of trajectories or other tracking results. The comparison between the two types of architecture is illustrated in Figure 2.27.

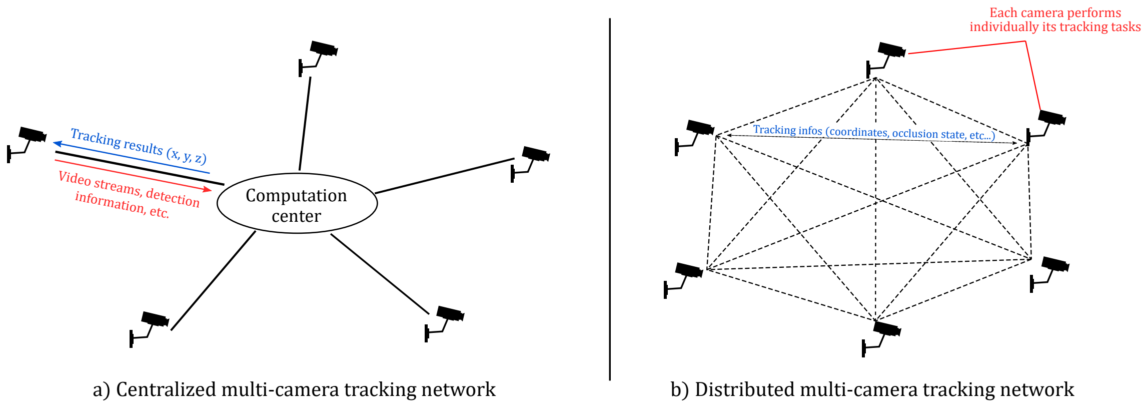

Tracking results (x, y, z)
Video streams, detection information, etc.
Computation center
a) Centralized multi-camera tracking network
Each camera performs individually its tracking tasks
Tracking infos (coordinates, occlusion state, etc...)
b) Distributed multi-camera tracking network

Figure 2.27 – Comparison between two main cameras network architectures. a) Centralized multi-camera tracking network, which tracking algorithm perform at the computation center, meanwhile b) Distributed multi-camera tracking network dispense tracking tasks for all cameras

The biggest difference between the centralized and distributed systems is that the centralized architecture is favorable for the complex multi-camera tracking with the eventual purpose to store the tracking information for *monitoring, crime investigation, people search*, etc. This multi-camera system requires the permanent data transferring from each camera to the central computer; as a result, this camera system requires a powerful computing center, high-speed connections to all cameras, a data storage, and more energy eventually. Contrarily, a distributed camera system aims to improve and enhance the tracking results on each distributed cameras. Each camera has its own tracking results, might occasionally demand its neighbors for help in case of tracking become difficult (occlusions, lost identities). Furthermore, the connection between distributed cameras do not need a high bandwidth and could be temporarily interrupted without affecting the whole tracking system, this leads to the efficiency of the use in energy and network bandwidth, which are critical for any distributed system. In comparison with the conventional centralized system, a distributed system has many significant advantages in practice, such as avoiding legal issues relative to storing private information, light-weighted system, simple to set up. Finally, this distributed system contributes to the novel concept of Smart Cameras [194]. In the following sections, we discuss in detail the tracking algorithms used in multi-camera networks.

2.5.2 Generic multi-view tracking

In multiple cameras tracking literature, the multi-camera systems can be categorized according to the camera topology: *non-overlapping* and *overlapping* systems. According to a survey of camera networks conducted by Radke [171] in 2010, the relationship between the overlapping camera network is modeled by an undirected graph in which an edge appears between 2 cameras if they observe some of the same scene points from different

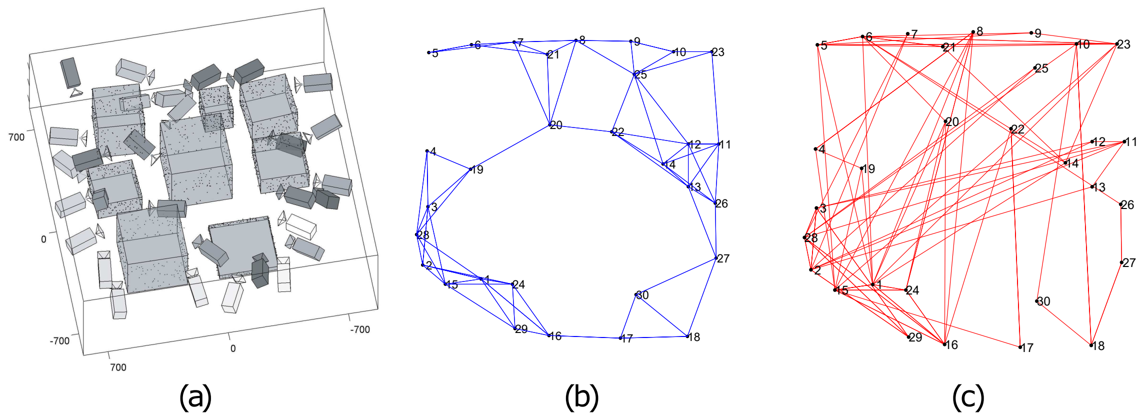


Figure 2.28 – (a) A simulated camera network (the focal lengths of the cameras have been exaggerated).(b) The corresponding communication graph, assuming each camera has the same fixed antenna range. (c) The corresponding vision graph. [171]

perspectives. Meanwhile, the cameras within a non-overlapping system are related via the likelihood of the event that an object in one camera appears in another after some amount of time. Furthermore, the camera network is modeled by a directed graph with expected transition probabilities. A camera network model is illustrated in Figure 2.28. With the non-overlapping network, the objective is to link the targets belonging to the same identities which appear in the videos of multiple cameras networks. This type of problem has recently been reformulated as *person re-identification* problem. However, the initial problem of non-overlapping cameras is addressed under the strict *spatio-temporal* constraint, for example, an individual cannot appear at the same time at different places in the camera network.

On the other side, the overlapping camera system is developed to resolve the most basic purpose of visual tracking algorithms, which is to find targets' full trajectories, then increase the accuracy, reliability, and robustness of a tracking system. Indeed, dealing with occlusion is still the unanswered question for any single view tracking algorithm. Even though several methods have the ability to recatch lost identities after occlusion, their complete trajectories will never be determined within one single view system. In many tracking applications that serve to analyze the behavior of crowds or to detect abnormal movements of people, observing full trajectories is essential. This type of tracking problem is not necessarily under the same constraints as mentioned in the non-overlapping topology, but it essentially requires the calibration and synchronization of all cameras in the network. We emphasize that in the literature on multiple camera tracking system, the term *multi-view tracking* means tracking in an overlapping camera system, whereas the term *multi-camera tracking* is more general, it can be neither overlapping or non-overlapping system. In terms of camera calibration, calibration helps a single-camera project its tracking result on the common plane to obtain the targets' coordinates in order to collaborate with other cameras inside the system. These coordinates from all cameras might be useful if all of them are gathered at the same time; i.e., this requires synchronization between cameras. More concretely, calibrating a camera is to determine the mapping function that maps pixel coordinates on images to 3D coordinate on the real world. This can be done through homographic transformation matrices after correcting the image deformations caused by flaws on the lenses of camera [206, 245, 117]. A survey on overlapping camera networks by Taj and Cavallaro [199] summarized the state-of-the-art approach until 2009. The majority of multi-view tracking approaches are inspired by those of remote sensing. The diagram of these multi-view tracking methods is presented in Figure 2.29.

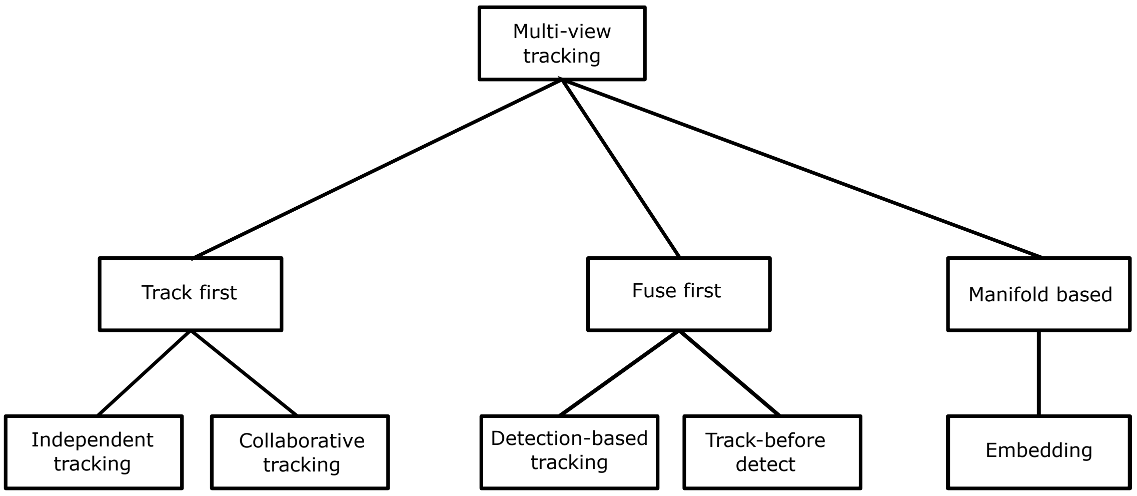


Figure 2.29 – Overview of the classic multi-view approaches [199].

Following this categorization, the track-first approaches perform tracking on each camera, then project and link the tracking results on the other cameras. In this branch, several methods perform multiple single object trackers simultaneously, then fuse their results. The fusion step broadly rectifies the results on each camera and infers the final trajectories on the ground plane. The typical papers using this approach include Kalman filter by Black et al. [24], Bayes tracker by Cai, and Aggarwal [33]. Some papers extend this approach while collaborating trackers such as graph matching approach [5], Gaussian Mixture Probability Hypothesis Density (GMPHD) approach [166] or providing a multi-object trackers such as collaborative particle filters [71, 170, 65]. One of the advantages of this approach is only a little amount of information being transferred inside the camera network. Whereas, the fuse-first approaches are usually seen in the Detection-based tracking (or *Tracking-by-Detection*) algorithms [159, 5, 120, 73, 124, 81, 20, 50, 3].

A typical algorithm in this approach is the Multi-camera people tracking with a Probabilistic Occupancy Map by Fleuret et al. [81]. In detail, the authors introduced a Probabilistic Occupancy Map (POM), which estimates the marginal probabilities of the presence of individuals on a fixed-size 2D grid map, given binary images corresponding to the result of a background-subtraction from different viewpoints. The detection step results in the binary image of background subtraction on each view, and then under the appearance model, they determine the possibilities of the presence of targets on the POM map. The illustration of the method is described in Figure 2.30. Indeed, the POMs obtained in the entire videos are modeled as the input of the directed graph of a dense network flow problem. Each node on the network flow represents a cell of the 2D grid map (POM). They stack all POMs and add the edges which connect a node to those neighbors in the next frame. Each path found in the network flow has a minimum total cost. The unary cost at a node is its value on POM, and the pairwise cost is the binary value indicating whether or not a node is inside its neighbor in the previous frame. The illustration is shown in Figure 2.31. The K paths of K individuals in the videos are computed via K *shortest Paths problem* [20]. There are many disadvantages to this approach. First, the method only works under the assumption that the number of targets K is known. Secondly, the scheme of this approach is impractical for online applications, because it processes through two main steps: fusing detections on the map, solving the K-paths problem to extract trajectories. Thirdly, a significant amount of data is being transferred during the tracking process. Finally, the accuracy of the tracking algorithm significantly depends on the grid size of the map. Lately, there have been a few efforts trying to improve the accuracy of this method, such as Deep Occlusion Reasoning by Baque et al. [13]. Having a similar idea of fusing detection first, Cocsar et al. [50] sparsely encode the detection information to minimize the size of data before sending.

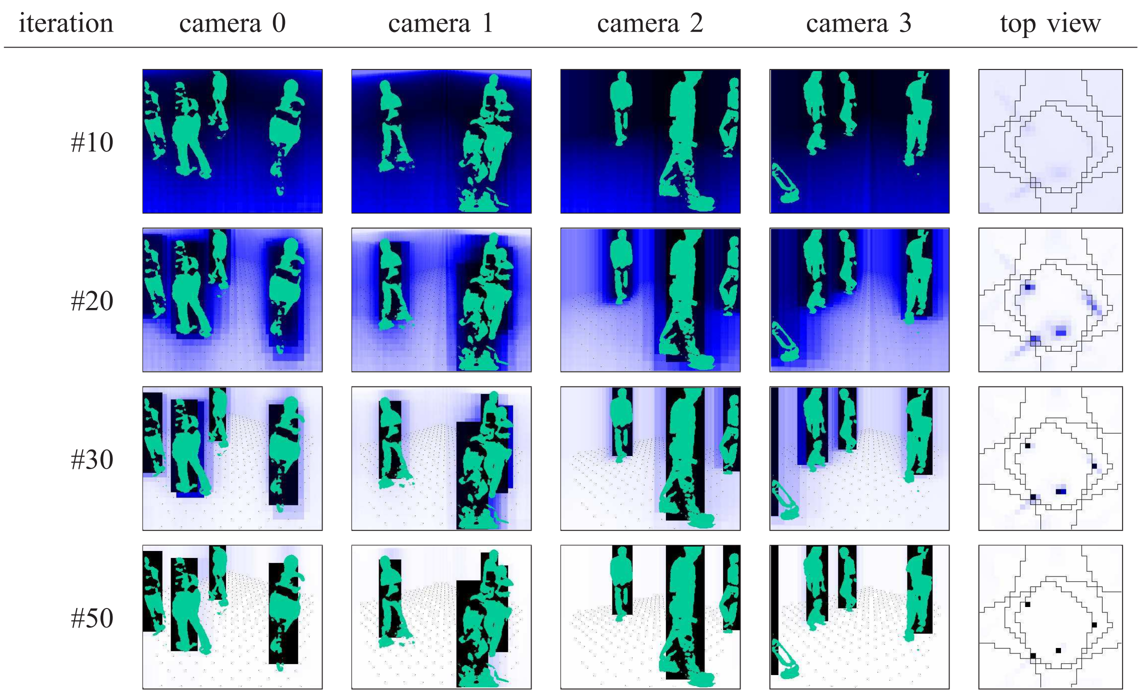


Figure 2.30 – The estimation of the Probabilistic Occupancy Map (POM) (the last column). Camera views show both background subtraction blobs and the synthetic average image corresponding to different iterations [81].

The last category in [199] is the manifold-based approaches that project features on a manifold to identify the evolution of the data, which includes the positions of targets in the case of the lack of the camera calibration data. Sunderrajan et al. [195] proposed a method for object tracking from different views based on multi-camera appearance modeling, which uses a manifold to model the *global multi-view appearance* for targets. Between any pair of views, the extracted features (e.g., the histogram of oriented gradients features (HOG) and normalized color features) from a target are used to create a manifold surface [84, 89]

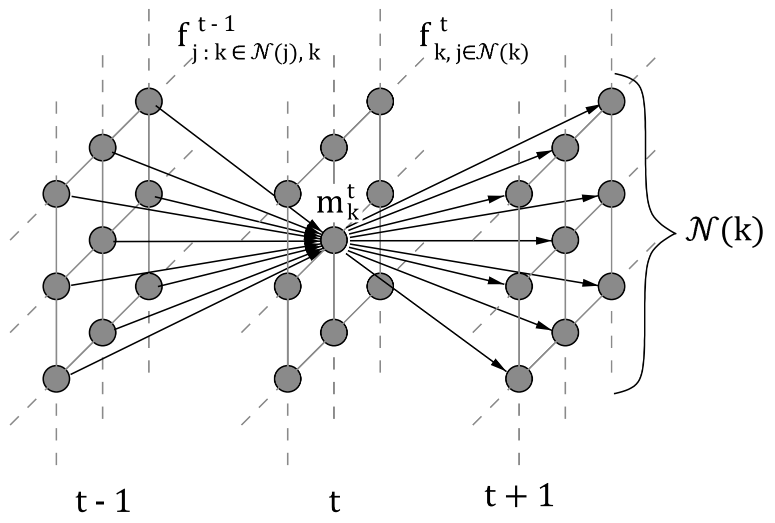

$f^{t-1}_{j:k\in\mathcal{N}(j),k}$
$f^{t}_{k,j\in\mathcal{N}(k)}$
m^t_k
$\mathcal{N}(k)$
t - 1
t
t + 1

Figure 2.31 – Network flow model used for tracking objects moving on a 2D grid, such as in pedestrian tracking. For the sake of readability, only the flows to and from location k at time t are printed [20].

where this features pair is two distinct points on it. Intuitively, the points represent the local appearance in single views of a particular target, and the manifold surface is its global appearance representation. The shortest path linking two points on the manifold is called the *geodesic*. On this geodesic, the authors interpret the intermediate points as the positive candidates, in addition to the negative candidates which are sampled around the target in all views, to learn a multi-view discriminative appearance classifier such as [6, 250, 7]. The illustration of the geodesic connecting two views of a target is presented in Figure 2.32.

To deal with the interactions between targets, e.g., mutual occlusion, the authors introduce interacting MCMC framework using the local and global particle filters with Markov Chain Monte Carlo (MCMC) sampling. For local particle filters, the observation likelihood is computed using the local appearance model and the object's interaction that are local to the camera. For global particle filters, the observation likelihood is computed based on global appearance models, multi-camera information, and scene priors.

Most of the above multi-view multi-object tracking algorithms aim to process tracking tasks online to adapt the requirement of many surveillance applications. However, there are many applications in which the online processing is not exigent such as crime investigation, analysis of crowd behavior. Hence, the data association approaches in MOT literature are straightforward and adaptable to extend in the context of multiple cameras. The next subsection is devoted to multiple cameras data association based methods.

2.5.3 Multiple cameras data association based methods

In this section, we focus on reviewing the Multiple Target Multiple Cameras approaches, which are extended or extendable from Multiple Object Tracking (MOT) or Multiple Target Tracking (MTT) on a single-view. Systems of multiple non-overlapping cameras, in reality, aim to observe and monitor people in a large area for security purposes. Because

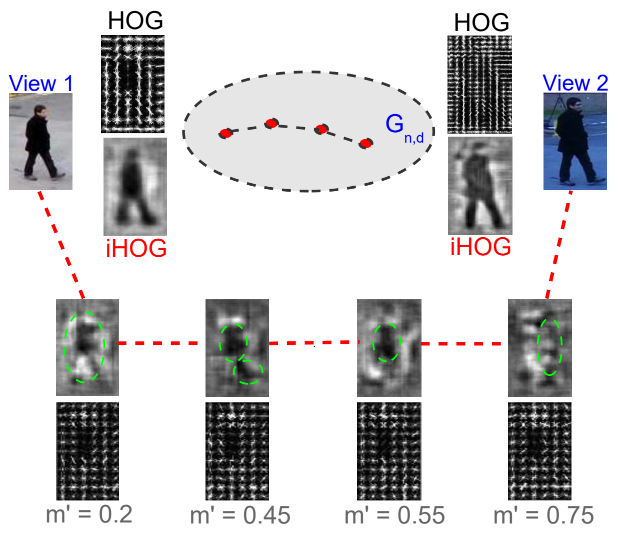


Figure 2.32 – Training samples for global appearance learning is obtained by projecting samples from the geodesic, which links view 1 to view 2, onto different generative subspaces obtained by varying m'. The eclipse represents the Grassmann manifold $G_{n,d}$ with S_1 and S_2 are points on it[195].

of the particularity of its objective, high-resolution cameras are usually set up at high positions to maximize their field of view, which rarely overlap each other, but they are not completely separated to avoid blind spots while surveilling on a large area. As mentioned in the Section 2.4.4, implementing the data association based MOT algorithm in a multiple camera context is obviously feasible, because the input of all those algorithms is simply time-labeled detections without knowing where they come from either a single camera or a number of cameras. However, it requires several changes in the graph model and the spatio-temporal constraints. For example, on the one hand, two disjoint cameras cannot be seeing the same individual at the same time, so the graph is not allowed to have an edge linking any two detections belonging to two disjoint cameras. On the other hand, the graph can build edges connecting the detections that are from different cameras but reside in the overlapping zones.

Solving the global optimization of the decomposition graph problem is computationally infeasible, so in practice, the global data association MOT algorithms such as graph multicuts [200, 179, 201, 202, 119], graph cliques [236, 64], network flow [237, 167, 240, 37] can perform on short subsequences of the videos from all cameras. Meanwhile, the other data association methods including bipartite matching [4, 191, 184, 224, 252, 232] and independent set [29, 123] do not address multi-camera tracking problem, because the tracklets are formed through the detections in consecutive frames (i.e., a short time window) of a single view, whereas tracking with multiple cameras is to connect trajectories of targets at different times.

Besides the global optimization approach for MTMC tracking, there is another approach that separates MTMC tracking into two steps: MOT on every single view, then linking the trajectories across cameras [204, 121, 5]. A typical method in this category is the

Constrained Dominant Set Clustering (CDSC) presented by Tesfaye et al. [204]. Given an edge-weighted graph $G = (V, E, w)$ (i.e., the unary cost is excluded), the goal of the algorithm is to find a subgraph that contains all or some of the elements of the constraint set, which forms a coherent and compact set for one individual. The method performs tracking in two stages. The first stage is to determine trajectories (called *tracklets* in the paper) of all targets in each camera, which is called *within-camera tracking* with the formulation of Constrained Dominant Sets. To enhance the tracking tasks to be able to manage the multi-entry/exit of targets, the authors proposed an additional data association step to cluster the tracklets belonging to the same identities. A tracklet mentioned in the paper [204] is a complete trajectory of an individual since it appears on the scene until getting out of the scene. The affinity between all considered detections is presented by an affinity matrix $A = (a_{i,j})$ where $a_{i,j}$ is the weight of the edge $w(i, j)$ connecting to two nodes (detections) i, j. Solving the CDSC problem via linear quadratic program, they obtain the clusters of trajectories on a single view, each cluster contains all motion history of an individual, called a *track* of each target. On the second stage, the algorithm links those tracks from all cameras, called *across camera tracking*. Similarly, a graph is built from tracks as its node, whereas its edges are defined by a corresponding affinity matrix depicting the similarity between those tracks. At this time, the across camera tracking stage releases the complete trajectories of targets on the whole camera network. The illustration is displayed in Figure 2.33.

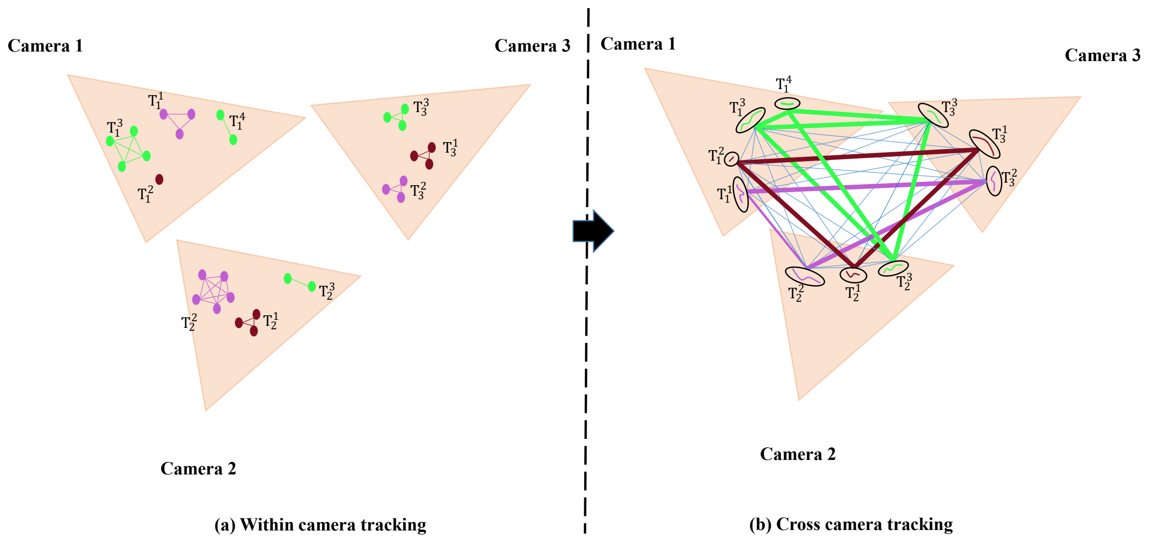


Figure 2.33 – A general idea of the Constrained Dominant Set Clustering (CDSC) method. (a) First, tracks are determined within each camera, then (b) tracks of the same person from different non-overlapping cameras are associated, solving the across-camera tracking. Nodes in (a) represent tracklets and nodes in (b) represent tracks. The i^{th} track of camera j, T_j^i, is a set of tracklets that form a clique. In (b) each clique in different colors represent tracks of the same person in non-overlapping cameras. Similar color represents the same person.[204]

Using the similar MTMC scheme, Ristani et al. [180] extended the Correlation Clustering method [179] on multiple view tracking. The author proposed an appearance feature learning method by finetuning a common CNN network, e.g., ResNet50 [99] under the re-identification scheme to maximize the ability to identify targets. The extracted features of detections are the input of the Correlation Clustering algorithm to cluster the targets' trajectories across cameras. Finally, the post-processing task is applied to remove the low-confidence trajectories and interpolate the missing detections. The overview of the method is illustrated in Figure 2.34.

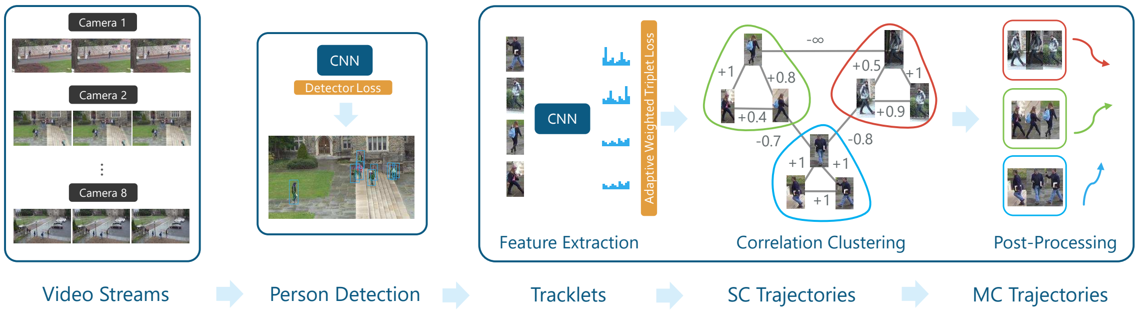


Figure 2.34 – An illustration of the Ristani et al. [180] approach. Given video streams, a person detector extracts bounding box observations from video. For trajectory inference, a feature extractor extracts motion and appearance features from observations. These are in turn converted into correlations and labeled using correlation clustering optimization. Finally, post-processing interpolates missing detections and discards low confidence tracks. Multi-stage reasoning repeats trajectory inference for tracklets, single- and multi-camera trajectories[180].

To categorize the MTMC tracking methods, we rely on particular aspects such as *online* or *offline* methods, *centralized* or *distributed* systems, multiple camera tracking strategy (global optimization vs two-step optimization). According to the computational model, there are two branches of multiple camera tracking methods: centralized algorithms and distributed ones. The majority of the MTMC tracking algorithms in the state-of-the-art are centralized algorithms [237, 141, 240, 204, 180, 200, 201, 202, 119, 237, 167, 240, 37, 236, 64]. These approaches are generally suitable for offline tracking. On the other hand, several algorithms have used a distributed computational model in an attempt to create a probability map (see *e.g.* [81, 168]), meanwhile, the other papers [170, 195, 132] developed the collaborative tracking frameworks to incorporate the tracking results across cameras. Unlike the centralized algorithms, distributed multi-camera tracking algorithms are online. To summarize the MTMC tracking methods, we classify them in the table 2.1.

Approaches	Methods	Appearance feature	Optimization	Online	Near-online/ Frame-batch	Offline	Multi-Camera/ MC extendable
Bipartite Matching	Andriluka et al. (2008) [4]	DPM	-	✓			
	Shu et al. (2012) [191]	DPM	Greedy bi-matching	✓			
	Sadeghian et al. (2017) [184]	CNN,LSTM	Hungarian algorithm	✓			
	Wojke & Alex (2017) [224]	CNN	Hungarian algorithm	✓			
	Zhu et al. (2018) [252]	CNN,LSTM	Heuristic bi-matching	✓			
	Xu et al. (2019) [232]	CNN,GRU	DHN	✓			
	Xu et al. (2019) [231]	STRN	Hungarian algorithm	✓			
Network Flow	Zhang et al. (2008) [237]	HC	ILP (MAP estimation)			✓	✓
	Pirsiavash et al. (2011) [167]	HC, HoG	ILP (KSP)			✓	✓
	Berclaz et al. (2011) [20]	POM	ILP (KSP)			✓	✓
	Zhang et al. (2015) [240]	HoG, BTF	ILP (KSP)			✓	✓
	Chari et al. (2015) [37]	IoU	ILP (KSP)			✓	✓
Graph multicut	Ristani and Tomasi (2014) [179]	HC (HSV)	BIP		✓	✓	✓
	Tang et al. (2015) [200]	DPM	ILP		✓	✓	✓
	Tang et al. (2016) [201]	DM	ILP			✓	✓
	Keuper et al. (2016) [119]	Optical flow	ILP		✓	✓	✓
	Tang et al. (2017) [202]	CNN	ILP			✓	✓
Graph Clique	Zamir et al. (2012) [236]	HC	RANSAC		✓	✓	✓
	Dehghan et al. (2015) [64]	HC	BIP		✓	✓	✓
	Tesfaye et al. (2017) [204]	CNN	PQP			✓	✓
Maximum Weight	Brendel et al. (2011) [29]	HC, HoG	ILP		✓		
Independent Set	Kim et al. (2015) (MHT) [123]	MORLS	ILP		✓		
	Choi (2015) (NOMT) [45]	ALFD	ILP		✓		

Table 2.1 – Data association based MOT methods in multiple cameras context. Abbreviation: DPM - Deformable Part Model [77]; LSTM - Long short-term memory [1]; CNN - Convolutional Neural Network [135, 99]; GRU - Gated Recurrent Unit [232]; DHN - Deep Hungarian Network [232]; STRN - Spatial Temporal Relation Network [231]; HC - Color histogram; HoG - Gradient histogram; IoU - Intersection over Union; POM - Probabilistic Occupancy Map [81]; BTF - Brightness Transfer Functions; MORLS - Multi-Output Regularized Least Squares [138]; ALFD - Aggregated Local Flow Descriptor [45]; ILP - Integer Linear Program; MAP - Maximum-a-posteriori; KSP - K-shorstest path; RANSAC - RANdom SAmple Consensus; BIP - Binary Integer Program; PQP - Parameterized Quadratic Program; MC - Multi-Camera.

2.5.4 The re-identification problem

In this section, we reformulate tracking as a re-identification problem. As a particular case of people tracking within non-overlapping cameras, person re-identification is considered as a post-processing step in a multi-camera tracking system. Because person re-identification has many real-life applications, e.g., crime investigation, check-in security, it has been studied intensively in the computer vision community over the last decade [247]. In the literature, person re-identification is placed after tracking and detections within the application pipeline of any camera system. Generally, Re-identification (Re-ID) is defined as a process of establishing a correspondence between images of a person taken from different cameras. Given an image or multiple images of an unknown person, called a *probe* (or *query*), and a *gallery* which is a set consisting of known people and junk images which represent the false positives from detections, the objective is to produce a ranked list of all the people in gallery based on their visual similarity with the *probe* (the unknown person). As a result, the highest-ranked match in the *gallery* will provide an ID for the unknown person, thereby identifying the *probe* [19]. Figure 2.35 illustrates an end-to-end person re-ID system.

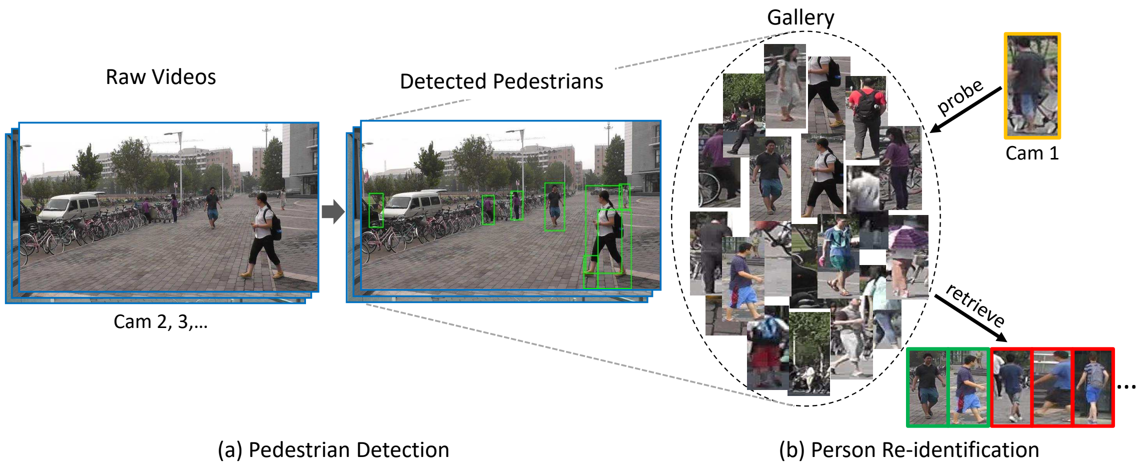


Figure 2.35 – The illustration of an end-to-end person re-ID system that includes (a) pedestrian detection and (b) re-identification[247].

In the Re-ID literature [247, 19], the Re-ID methods are categorized into two main approaches: hand-crafted and Deep learning re-ID approaches. A hand-crafted re-ID approach consists of two fundamental components: a *description* gathering appearance features of targets and a *distance metric* measuring the distance between appearance descriptions. As mentioned in the section discussing the appearance features for data association, appearance feature selections play a crucial role in identifying people. In human descriptions, the most relevant feature is color, while texture features seem less useful and rarely used. There are the enhanced features based on color and human body structure [76], for example, the weighted color histogram (WH) assigns larger weights to pixels near the symmetrical axis and forms a color histogram for each body part; and the maximally stable color regions (MSCR) detects stable color regions and extracts features, such as color, area, and centroid of these regions. Some methods enhance the RGB color model with CMYK and HSV models (Gray et al. [92], Mignon et al. [157], Das et al. [62]). In appearance feature hand-crafted re-ID approach, estimating the distance between two feature vectors via a good distance metric is essential. According to the survey conducted by Zheng et al. [247], in person re-ID, most papers used supervised global distance metric learning which learns a non-linear function that maps feature vectors from a current space to another where the mapping keeps same-class vectors gathered close and those of different-class stay far apart. The most commonly used distance metric is the Mahalanobis distance [126, 230, 220, 106], which is a measure of the distance between a point P (*probe*) and a distribution D (*gallery*). Figure 2.36 visualizes the results of projecting people (represented by feature vectors) into an embedding space. Some works are using the supervised local distance metric learning instead, such as David et al. [63].

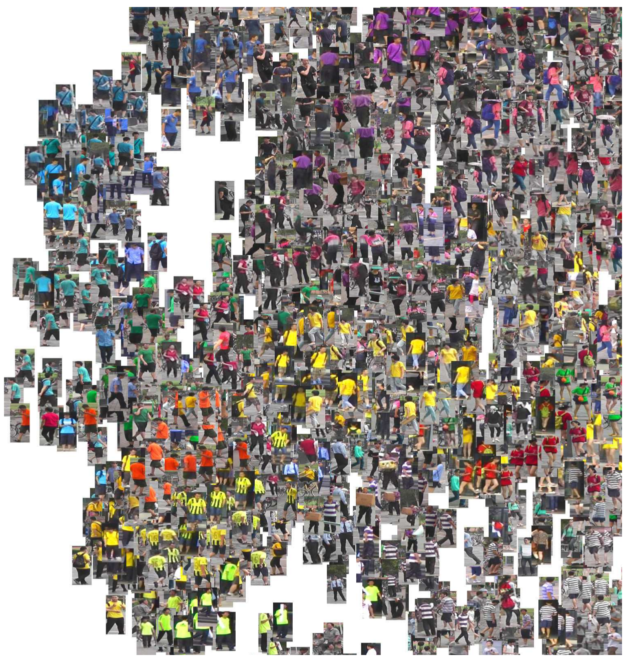


Figure 2.36 – Visualization of different identities on an embedding space. A small crop of the Barnes-Hut t-SNE [209] of the learned embeddings for the Market-1501 test-set [104].

Since deep learning has emerged as an effective technique to adaptively extract characteristic features, many works dedicated to using this technique in distance metric learning [187, 96, 94]. Indeed, deep metric learning is to learn a distance metric directly from raw input data by a deep neural network. Supervised learning in classification problems principally learns a model from labeled data to classify the testing data into the predefined labels in training datasets. However, metric learning, instead, is to learn a vector space in which the similarity between two data samples is measured by the distance between them, then on the learned metric space, we are able to cluster the similar samples in the same class, meanwhile to well separate the samples from different classes. Despite having good representations via larger and better deep neural networks, setting up constraints for "similar" and "dissimilar" pairs during training is another challenge to separate data without labels, i.e., *unsupervised learning.* Many works such as Li et al. [140] and Zhang et al. [244]

attempted to formulate this as *supervised learning*, e.g. multi-class classification problem. However, the accuracy in the testing phase deteriorated as the presence of "foreign" classes, not in the training dataset.

To alleviate this difficulty, pairwise losses have been introduced in [46, 192, 192]. Unlike the classification loss, which moves all samples to places with labels they belong to, optimizing pairwise loss tends to minimize the distance between pairs in the same class, i.e., *positive pairs*, while maximizing the distance between pairs in different classes, i.e., *negative pairs*. The setback of this method is that the algorithm optimizes positive pairs independently from negative pairs. With the purpose of making the training more efficient, [220] proposed the "triple" constraint that given a sample called *anchor*, the distance from this anchor to a similar sample should be smaller than to a dissimilar sample. Therefore, the triple loss aims to separate positive pairs from negative pairs by a *margin* [220, 169]. Nonetheless, methods with either pairwise or triple loss have to select pairs or triples to create a training sample. Consequently, they have a complexity of $O(n^2)$ or $O(n^3)$, respectively, with the number of samples n. To solve this issue, the author of the paper [169] proposed the normalized Softmax and SoftTriple losses, which elevate the sampling phase causing the expensive computational cost. There are many other methods with different losses, such as proxies [160] or class centroids [68].

As an effort to improve the accuracy of Re-ID task during the testing phase, some papers by Barbosa et al. [15] and Zhong et al. [251] introduced *hard training examples* which are typically generated by creating data augmentation. For clarity, we should mention that in re-ID learning approaches, *hard training examples* are the examples gathered from the different identities but having relatively the same appearance. Zheng et al. [248] used Generative Adversarial Network (GAN) for mining difficult examples. In [227], the author proposed a method to sample hard training data. [216] introduced the angular loss which finely "mines" hard training samples based on the angle between anchor, positive and negative simple on an embedding space. An extensive experimental study conducted by Almazan et al. [2] in 2018 on re-ID methods recommended good practices to achieve better accuracy in person re-ID.

2.6 Benchmark and performance measure

In this section, we only discuss the benchmarks for MOT and MTMC algorithms. We are aware that the Single Object Tracking community also uses another benchmark [228] to evaluate SOT trackers, but this is not the main focus of this thesis.

Evaluating Multiple Object Tracking algorithms is a sophisticated and complex task. Unlike the Single Object Tracking problem, MOT benchmarks have to take into account the errors caused by the interactions between objects. For example, an object can suddenly appear in the middle of a video, be hidden by other objects, which are also under the tracking process, and then reappear. As a result of different errors generated in different particular situations, a variety of indicators are being deployed in the MOT evaluation process. Summarizing all these indicators, MOT benchmarks define the universal scores to grade each tracker to produce a more general look on methods. However, it is not surprising that there are many benchmarks considering the success of the MOT tracking process under different aspects. Hence, a benchmark can be biased for several tracking purposes. In principle, choosing one single performance measure that suffices for all end-users with different needs is problematic.

In the state-of-the-art, Bernardin and Stiefelhagen [22] firstly released the CLEAR[1] metric for evaluating Multiple Object Tracking in a single camera. Since then, it has become a standard metric in the MOT community. In 2015, Leal-Taixé et al. [134] introduced MOTChallenge 2015[2] which is a Multiple Object Tracking benchmark. MOTChallenge 2015 includes a large collection of datasets, detections for all the sequences in datasets, a common evaluation tool providing several measures and a website to summit the tracking results of all MOT algorithms, for a fair comparison. The collection of datasets is ranging from *static* camera sequences to *moving* camera sequences; it is subdivided into many subsets of data for specific tasks such as 3D tracking, surveillance, sports analysis. All these data subsets possess a training dataset with both detection and ground-truth of sequences, a testing dataset with only detections provided. After using training data to tune the parameters of the MOT algorithms, the authors test their algorithms on testing data. Then they are encouraged to submit their tracking results on MOTChallenge website[3], the evaluation results will be published later. In the meantime, Ristani et al. [178] developed ID-measure in addition to a large datasets[3] which are the videos recording the campus of the University of Duke which are also divided into training and testing sequences. This performance measure is suitable for both single or multiple cameras in general.

2.6.1 Multiple Object Tracking metric in single-camera and multi-camera

In general, a MOT benchmark consists of two main steps: matching tracker hypotheses (i.e., *computed identities*) with ground-truth objects (i.e., *true identities*) (illustrated in Figure 2.38 (a)) in *each frame* and computing the matching score based on tracking hypothesis-vs-ground-truth trajectories on the *entire video sequence* (illustrated in Figure 2.38 (b)). In terms of the matching procedure, thresholding the hypotheses around the ground-truth objects excludes some *false positives* out of the matching process (Figure 2.37). In 2D evaluation, only image plane coordinates are involved, the hypothesis-object distance is evaluated by the Intersection over Union (IoU) ratio with a threshold of 0.5 [22]. In 3D evaluation, 3D world coordinates are used instead, and the positions of hypotheses and objects are only considered on a reference ground plane, i.e., the z-coordinate is always a constant and usually a zero-ground $z = 0$. The distance now is the Euclidean distance between two positions, in meter, with a threshold of $1m$ [22]. Finally, only the closest object-hypothesis pairs are matched, while all remaining objects and hypotheses are treated as *misses* and *false positives.*

For the scoring step, the MOT benchmark grades the hypothesis-vs-ground-truth trajectories matchings by accumulating the different error types defined as follows:

- f_{p_t} is the number of *false positives* generated by the tracking algorithm at frame t, and $FP = \sum_t f_{p_t}$ is the total false positive.

[1]CLEAR standing for Classification of Events, Activities, and Relationships
[2]https://motchallenge.net/
[3]http://vision.cs.duke.edu/DukeMTMC/details.html

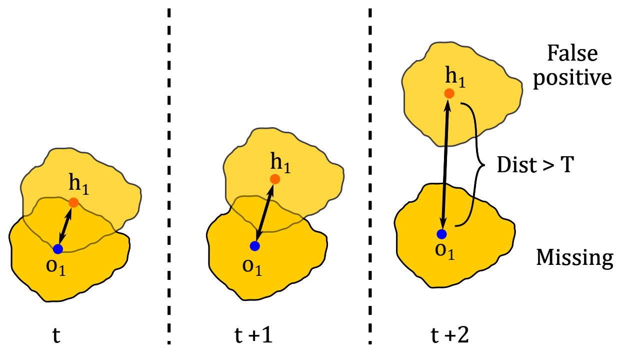


Figure 2.37 – A threshold T deciding whether the hypotheses are false positives or not. The distance between the hyothesis h_1 and the object o_1 exceeds the threshold at frame $t+2$ resulting in a *false positive* and a *miss* (i.e., false negative) [22].

- f_{n_t} is the number of true targets missed at frame t i.e., *false negative*, and the total false negatives is $FN = \sum_t f_{n_t}$.
- t_{p_t} is the number of true positive detections at time t, and the total true positive TP is $TP = \sum_t t_{p_t}$.
- *Fragmentation* occurs in frame t if a tracker switches the identity of a trajectory in that frame, but the corresponding ground-truth identity does not change. The number of fragmentations at frame t is ϕ_t, and the total fragmentation error $\Phi = \sum_t \phi_t$
- *Merge* happens when trackers merge two different ground truth identities into one between frames t' and t. The number of merges at frame t is γ_t, and the total merge error $\Gamma = \sum_t \gamma_t$
- *Mismatch* is either fragmentation or merge $\mu_t = \phi_t + \gamma_t$ and the total mismatch is $M = \sum_t \mu_t$

Figure 2.38 shows the different types of errors in MOT benchmark. In the single view multi-object tracking benchmark, based on the error types defined above, the Multiple Object Tracking Accuracy (MOTA) is defined as follows:

$$MOTA = 1 - \frac{FN + FP + M}{T}. \tag{2.2}$$

MOTA penalizes detection errors ($FN+FP$) and mismatches (M) normalized by the total number T of true detections. In principle, MOTA shows the frequency of making mistakes of a tracker in terms of misses, false positives, mismatches, failures to recover tracks, etc.

The metric has become popular and been widely adopted by the MOT community to measure the performance of MOT trackers. To extend the MOT metric in the context of multi-camera tracking, in the paper of Cao et al. [35] and Ristani et al. [178], these authors consider only the continuity of tracking between cameras. For example, an individual appears in the FOV of 2 different cameras at different times, the correct links

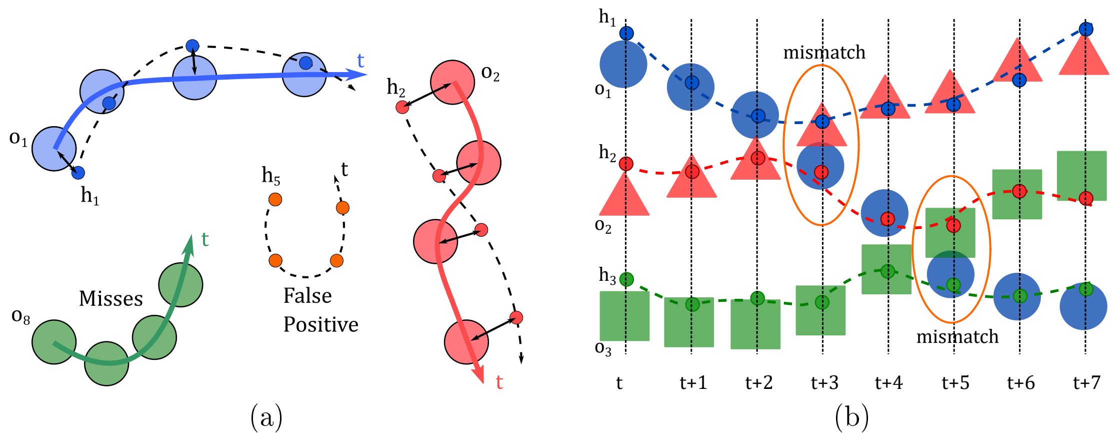


Figure 2.38 – Illustration of the different error types in MOT metric [22]. (a) Mapping tracker hypotheses to objects shows *false negatives* (misses), *false positives* and the correct tracking hypotheses i.e., *true positives*. (b) Mismatch error

between the trajectories of this individual from a camera to other measures the ability of re-identifying this person across cameras regardless of the camera configurations (*overlapping* or *non-overlapping*), which are called the *handovers* [178]. Cao et al. [35] proposed the Multi-Camera Object Tracking Accuracy (MCTA) score, which considers all multi-cameras aspects in terms of errors:

$$MCTA = \underbrace{\frac{2PR}{P+R}}_{F_1} \underbrace{\left(1 - \frac{M^w}{T^w}\right)}_{\text{within camera}} \underbrace{\left(1 - \frac{M^h}{T^h}\right)}_{\text{handover}}. \tag{2.3}$$

This score is the multiplication of the F_1 score (where P is the precision, and R is the recall), a "within camera" term which penalizes within-camera identity mismatches (M^w) normalized by true within-camera detections (T^w) and a "handover" term which takes into account wrong identity handover mismatches (M^h) normalized by the total number of handovers. Notice that the upper script w and h indicate the "within-camera" and "handover". However, Ristani et al. [178] pointed out that there are several issues relative to the ability to preserve identities of targets. The CLEAR-MOT metric rates the MOT trackers, which continuously following targets higher than those which might switch the identities of targets multiple times to conserve their initial identities. This metric is appropriate to evaluate the tracking systems, which prefer to observe events to analyze further group behaviors (i.e., *event-based* measures). Meanwhile, for the users who are interested in the applications, including sports, security, or surveillance, preserving identities of targets is vital, it motivated the authors [178] to develop a novel *identity-based* measure (called *ID-measure*) to fulfill the need.

To focus on conserving identities, the authors propose to measure the performance of a tracking algorithm not by how often mismatches occur, but by how long the algorithm correctly identifies targets. In detail, a bipartite graph $G = (V_T, V_C, E)$ is built from two sets of nodes V_T and V_C, and the edges between them as follows:

- V_T includes one "regular" node τ for each true trajectory and one "false positive" node f_γ^+ for each computed trajectory γ.
- V_C includes one "regular" node γ for each computed trajectory and one "false negative" node f_τ^- for each true trajectory τ.
- A set of edges connect each node on V_T to only one node on V_C. Therefore, every (τ, γ) match is a True Positive ID (IDTP). Every (f_γ^+, γ) match is a False Positive ID (IDFP). Every (τ, f_τ^-) match is a False Negative ID (IDFN). Every (f_γ^+, f_τ^-) match is a True Negative ID (IDTN) (see the Fig. 2.39).

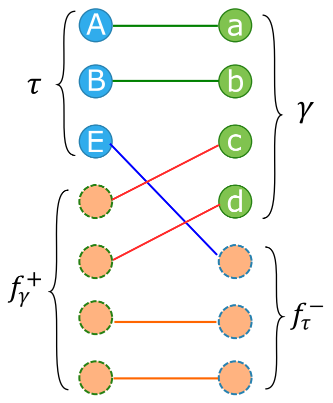


Figure 2.39 – Illustration of Matching the true trajectories (blue nodes) and computed trajectories (green nodes) at a single frame, The process operates on all frames to compute False Positive ID (IDFP), False Negative ID (IDFN), and True Negative ID (IDTN).

The mapping procedure occurs under a matching condition when the distance between a computed target and a true identity has to be smaller than a thresholding value Δ. Unlike the CLEAR-MOT metric of Bernardin and Stiefelhagen [22], ID-measure uses the *binary score* $m \in \{0, 1\}$ for every error values, then the one-to-one matching is done by minimizing the cumulative false positive and false negative errors, and the overall cost is the number of misassigned detections for all types of errors. In other words, this cumulative error cost increments one every time an error, e.g., miss, false negative, is found. After finishing the mapping process, based on the optimal Bipartite Matching, the IDFP, IDFN, IDTP scores are consecutively the sums of the cumulative false positive, false negative and true positive over all matched ground-truth trajectories and matched computed trajectories.

2.6.2 MOT and MTMC Datasets

In terms of Multi-Target Multi-Cameras (MTMC) tracking datasets, we aim to collect the multiple view video sequences with camera calibration data. Therefore, most of the dataset collection of the MOTChallenge 2015[4] is inadequate in our case study for many reasons such as moving camera videos, single view sequences, non-availability camera calibration information, or the asynchronization between cameras. As following, we list all the MTMC datasets with calibrated and synchronized cameras found in the current literature.

[4]https://motchallenge.net/

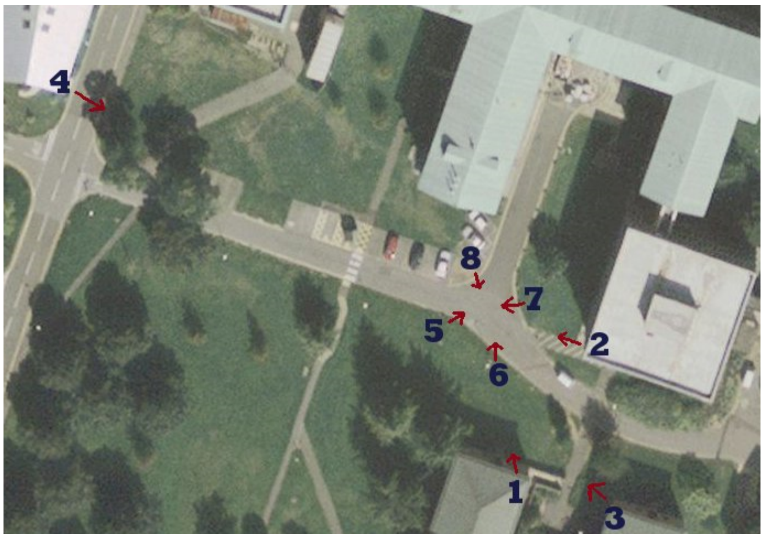


Figure 2.40 – Topology of camera network in the PETS 2009 datasets[78].

Firstly, the PETS2009 datasets by Ferryman and Shahrokni [78] are the most well-known dataset in the Multiple Object Tracking community in both single and multiple cameras contexts. In detail, the PETS2009 datasets are multi-camera sequences containing different crowd activities. The video sequences are made up of three types of crowd surveillance characteristics/events within an outdoor scene. Many different scenarios are made by multiple overlapping cameras filming approximately forty actors at a road junction. The topology of these cameras is shown in Figure 2.40. More specifically, the challenge includes the estimation of crowd person count and density, tracking of individuals within a crowd, and detection of flow and crowd events. All sequences are separated into three datasets: the training dataset (Dataset S0) provides the background information in all views which is in favor of background subtraction algorithms to detect people; the dataset S1 is for person count and density estimation; the dataset S2 aiming to tracking applications consists multiple sequences with different crowded level; and the dataset S3 is used for Flow Analysis and Event Recognition. The videos are made by two different types of cameras; one is more sensitive to the blue color frequency than another. Secondly, the Multi-camera

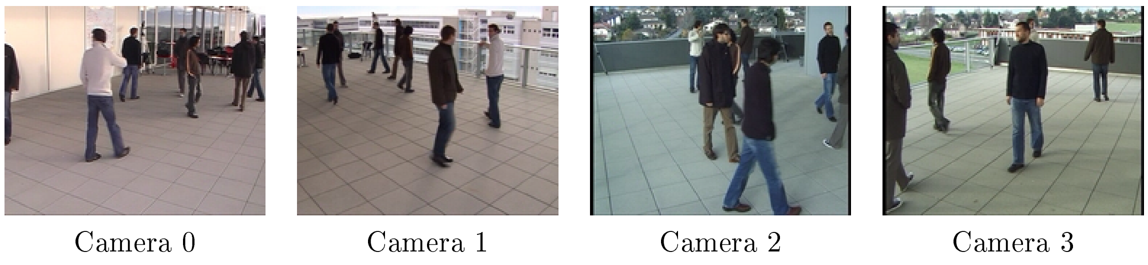


Figure 2.41 – The Terrace sequences in Multi-camera pedestrians videos EPFL dataset. The sequences are made in outdoor scene by 7 people in front of 4 DV cameras, for around 3 1/2 minutes.

pedestrians videos[5] from a vision project of EPFL[6] were presented in the MOT tracking paper of Berclaz et al. [20] as the testing dataset for their detection and tracking framework. The video sequences are realized by multiple overlapping and synchronized cameras filming the same area with different angles. The camera calibration information is also provided. These multi-camera video sequences consist of Laboratory sequences, Campus sequences, Terrace sequences, Passageway sequence, and Basketball sequence.

As a reference MTMC tracking dataset of the ID-measure performance evaluation for Multi-Target Multi-Camera Tracking algorithms, Ristani et al. [178] released a large MTMC tracking dataset more than 2 million frames and more than 2,700 identities. It consists of 8×85 minutes of 1080p video recorded at 60 frames per second from 8 static cameras deployed at the Duke University campus during periods between lectures when the pedestrian traffic is heavy. The cameras are located at different places on the campus, and they are mainly non-overlapping. Calibration data determines homographies between images and the world ground planes. All trajectories were manually annotated by using an interface they developed to mark trajectory key points and to associate identities across cameras. Comparing to the existing MTMC datasets, this is the largest dataset of multiple non-overlapping cameras filming in a real-world environment. The position of the camera network in Duke MTMC dataset is visualized in Figure 2.42.

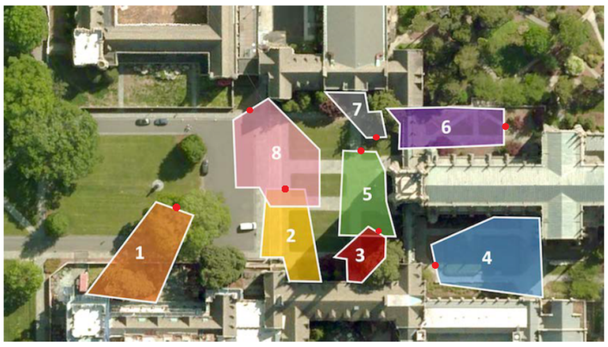


Figure 2.42 – Topology of the 8 cameras in the Duke MTMC dataset[180]. The red spots indicate the view direction of cameras.

[5]https://cvlab.epfl.ch/data/data-pom-index-php/
[6]Ecole polytechnique fédérale de Lausanne

Chapter 3

Sparse coding for collaborative tracking mono-object in multi-view

In this chapter, we present our first contribution to implementing particle-filter-based single object visual tracking methods on multiple camera systems. The tracking method is based on the state-of-the-art particle framework, which is commonly used in most of the Single Object Tracking (SOT) algorithms. In those SOT methods developed on single view, the problem of occlusion arises while a targeted object moves in a complex environment, and its entire movement cannot be observed and tracked fully. In various circumstances of wild tracking videos, this is a typical unsolvable problem on single view tracking. This raises the necessity of using multiple views to keep tracking occluded targets, which is impossible in a single view setting. Our work focuses on implementing SOT methods in a multiple view setting in order to solve the occlusion problem. Our approach relies on the standard particle filter framework, which is presented in detail in this chapter (Sec. 3.1). To track targets in each view, we deploy the sparse representation to model target through tracking procedure, which is described in Section 3.2. The following is the list of notation used in this chapter:

$\mathbf{x}_t$	The hidden state at time t
$\mathbf{y}_t$	The observation at time t
$\mathcal{X}$	The state space of $\mathbf{x}_t$
$\mathcal{Y}$	The observation space of $\mathbf{y}_t$
$\mathbf{x}_{0:t}$	The sequence of hidden states $\{\mathbf{x}_0, \dots, \mathbf{x}_t\}$ from 0 to t
$\mathbf{y}_{0:t}$	The sequence of observations $\{\mathbf{y}_0, \dots, \mathbf{y}_t\}$ from 0 to t
$p(\mathbf{x}_0)$	The initial distribution of the process' states
$p(\mathbf{x}_t \vert \mathbf{x}_{t-1})$	The transition model
$p(\mathbf{y}_t \vert \mathbf{x}_t)$	The marginal distribution or the likelihood distribution
$p(\mathbf{x}_{0:t} \vert \mathbf{y}_{1:t})$	The posterior distribution of the sequence of states given a sequence of observations up to time t
$p(\mathbf{x}_t \vert \mathbf{y}_{1:t-1})$	The prior distribution of the state at t given a sequence of observations up to time $t-1$

$\left\{\mathbf{x}_{0:t}^{(n)}\right\}_{n=1}^{N}$	The particles to estimate a distribution
$\left\{w_t^{(n)}\right\}_{n=1}^{N}$	The weights of particles
$P_N(d\mathbf{x}_{0:t}\|\mathbf{y}_{1:t})$	The empirical estimate of the distribution $p(\mathbf{x}_{0:t}\|\mathbf{y}_{1:t})$ via N particles
$d\mathbf{x}_{0:t}$	The small quantized region at $\mathbf{x}_{0:t}$
$\delta_{\mathbf{x}_{0:t}^{(n)}}(d\mathbf{x}_{0:t})$	The delta-Dirac mass located at $\mathbf{x}_{0:t}^{(n)}$
$\mathbf{y}^{(n)}$	The candidate of the n^{th} particle
$\mathbf{D}$	The dictionary of the target
$\mathbf{d}_i$	The target template of the dictionary
$\mathbf{t}_i$	The trivial template of the dictionary
$\mathbf{a}$	The coefficient vector
C_k	The k^{th} camera in the camera network
$s_{t,k}$	The sparsity level of targets at frame t on camera C_k
$p_k(\mathbf{y}_t\|\mathbf{x}_t)$	The likelihood distribution on camera C_k
m_k	The value of update timer on camera C_k

3.1 Particle filter framework

Particle filter is a well-known technique of *Sequential Monte Carlo* methods [70] that are used to approximate the unknown states of a process from the sequentially given observations. In applications, the states of processes are modeled as Markovian, non-linear, non-gaussian state-space models. In the visual tracking literature, the position of objects is estimated from a given series of images captured by visual sensors, e.g., cameras. Therefore, the movement of an object is modeled in the temporal and spatial space by a non-linear process whose states are the object's position evolving in time. Let $\{\mathbf{x}_t; t \in N\}, \mathbf{x}_t \in \mathcal{X}$ denote the hidden states of a Markov process with an initial distribution $p(\mathbf{x}_0)$ and a transition probability $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. While $\{\mathbf{y}_t; t \in N^*\}, \mathbf{y}_t \in \mathcal{Y}$ denotes the observations which are conditionally independent given the process $\{\mathbf{x}_t; t \in N\}$ and the marginal distribution $p(\mathbf{y}_t|\mathbf{x}_t)$. The described Markov model is summarized as follows:

(The initial distribution)	$p(\mathbf{x}_0)$	
(The transition model)	$p(\mathbf{x}_t\|\mathbf{x}_{t-1})$	for $t \geq 1$
(The marginal distribution)	$p(\mathbf{y}_t\|\mathbf{x}_t)$	for $t \geq 1$

By convention, we denote $\mathbf{x}_{0:t} := \{\mathbf{x}_0, \ldots, \mathbf{x}_t\}$ and $\mathbf{y}_{1:t} := \{\mathbf{y}_1, \ldots, \mathbf{y}_t\}$, respectively, the sequences of the states and the observations up to time t. The ultimate goal is to estimate *recursively* in time the *posterior distribution* $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, its *marginal distribution* or *filtering distribution* $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. Given a time instance t, the posterior distribution is written through *Bayes's theorem.*

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t-1})}{\int_{\mathcal{X}} p(\mathbf{y}_t|\mathbf{x}'_{0:t})p(\mathbf{x}'_{0:t}|\mathbf{y}_{1:t-1})d\mathbf{x}'_{0:t}}, \tag{3.1}$$

$$\text{or} \quad p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t-1}). \tag{3.2}$$

Straightforwardly, this posterior distribution formula can be rewritten as a recursive form:

$$p(\mathbf{x}_{0:t+1}|\mathbf{y}_{1:t+1}) = p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})\frac{p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1}|\mathbf{x}_t)}{p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t})}. \tag{3.3}$$

Meanwhile, the marginal distribution $p(x_t|y_{1:t})$ is only computed *recursively* in two steps: *predicting* the prior distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ and *updating* the marginal distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. The prior distribution is predicted from the marginal distribution at time $t-1$ by the following formula:

$$\underline{\text{Prediction}}: \qquad p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int_{\mathcal{X}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) \cdot p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1}. \tag{3.4}$$

The marginal distribution at time t is calculated from the obtained prior distribution.

$$\underline{\text{Updating}}: \qquad p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{\int_{\mathcal{X}} p(\mathbf{y}_t|\mathbf{x}'_t)p(\mathbf{x}'_t|\mathbf{y}_{1:t-1})d\mathbf{x}'_t}, \tag{3.5}$$

$$\text{or} \qquad p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1}). \tag{3.6}$$

In SOT tracking algorithms, it suffices to determine only the marginal distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

According to the particle filter framework in tracking algorithms, the movement of a object is described by a hidden Markov model as follows:

$$\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, \mathbf{v}_t), \tag{3.7}$$

$$\mathbf{y}_t = h_t(\mathbf{x}_t, \mathbf{w}_t), \tag{3.8}$$

where $\mathbf{x}_t$ represents the hidden state at time t, $\mathbf{y}_t$ the measurement or observation, f_t is the transition function of $\mathbf{x}_t$, h_t is the measurement function, and $\mathbf{v}_t$ and $\mathbf{w}_t$ are independent white noises. The initial Markov model at the beginning of the section is now being transformed to adapt to tracking problem. Indeed, the transition model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is a deterministic function $f_t(.)$ with the inputs including the last state $\mathbf{x}_{t-1}$ (markovian property) and a independent noise $\mathbf{v}_t$ (randomness). The observation function $h_t(.)$ is the mapping of the current state $\mathbf{x}_t$ and a random noise $\mathbf{w}_t$ to the observation space $\mathcal{Y}$.

In the classic object tracking problem using the particle filter framework, the hidden state $\mathbf{x}_t$ represents the pixel position of the target, being aware that various representations of the hidden state have been proposed in the literature. As a textbook example, Ross et al. [181] use $\mathbf{x}_t = (x_t, y_t, s_t, \alpha_t)$ where x_t and y_t are the 2-D coordinates of (the center of) the target, s_t is the scale parameter of the bounding box and α_t is the aspect ratio at time t.

Computing the integrals in the formulations (3.4) and (3.5) in the continuous domain is impractical, because of the difficulties to determine the prior and posterior distributions. However, in numerical computation, those distributions can be estimated via Monte Carlo sampling. The particle filtering framework for tracking simulates N independent and identically distributed (i.i.d) random samples, named as ***particles***, $\left\{\mathbf{x}_{0:t}^{(n)}\right\}_{n=1}^{N}$ according to $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. An empirical estimate of the distribution is given via Monte Carlo sampling:

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \approx P_N(d\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) = \frac{1}{N}\sum_{n=1}^{N}\delta_{\mathbf{x}_{0:t}^{(n)}}(d\mathbf{x}_{0:t}), \tag{3.9}$$

where $d\mathbf{x}_{0:t}$ means the small quantized region at $\mathbf{x}_{0:t}$ and $\delta_{\mathbf{x}_{0:t}^{(n)}}(d\mathbf{x}_{0:t})$ denotes the delta-Dirac mass located at $\mathbf{x}_{0:t}^{(n)}$.

Nonetheless, in practice, tracking an object is to estimate a set of object's positions from the beginning up to the time t. Therefore, the dimension of the states to be estimated $\mathbf{x}_{0:t}^{(n)}$ grows dramatically with time and leads to the unfeasible computation. To solve this problem, the perfect Monte Carlo (MC) sampling is replaced by a sequential MC approach, named Sequential Importance Resampling (SIR). Hence, our aim now is to approximately determine the marginal distribution of the current state given the observations up to current frame $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

As described in [70], the hidden states of a target including its positions are defined via the marginal distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ which is approximated by a weighted sum of N Dirac masses $\delta_{\mathbf{x}_t^{(n)}}(d\mathbf{x}_t)$ of a set of particles $\left\{\mathbf{x}_t^{(n)}\right\}_{n=1}^{N}$ of the state $\mathbf{x}_t$. The formulation of the distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is rewritten as follows:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx P_N(d\mathbf{x}_t|\mathbf{y}_{1:t}) = \sum_{n=1}^{N} w_t^{(n)} \delta_{\mathbf{x}_t^{(n)}}, \tag{3.10}$$

where $\mathbf{x}_t^{(n)}$ is the n^{th} particle and $w_t^{(n)}$ is its weight which is calculated via a target-templates comparison. In other words, the particles representing the coordinates of a target are generated around its current location in order to cover its most potential positions in the next frame.

Given an approximation of the distribution at time $t-1$, $P_N(d\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ and the observation likelihood $p\left(\mathbf{y}_t|\mathbf{x}_t\right)$, a bootstrap filter is implemented to estimate the distribution at time t, and it comprises three main steps:

1. The *diffusion step* consists in resampling a new set of unweighted particles $\left\{\mathbf{x}_t^{(n)}\right\}_{n=1}^{N}$ from the empirical prior distribution $P_N(d\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ which is the weighted particles set $\left\{\mathbf{x}_{t-1}^{(n)}, w_{t-1}^{(n)}\right\}_{n=1}^{N}$.

2. The *weight-computing step* then computes new particle weights given the new observation $\mathbf{y}_t$, and normalize them:

$$w_t^{(n)} \propto w_{t-1}^{(n)} p\left(\mathbf{y}_t|\mathbf{x}_t^{(n)}\right), \text{ such that } \sum_{n=1}^{N} w_t^{(n)} = 1. \tag{3.11}$$

3. The *update step* is to add the weights $\left\{w_t^{(n)}\right\}_{n=1}^{N}$ to the unweighted particles, $\left\{\mathbf{x}_t^{(n)}\right\}_{n=1}^{N} \rightarrow \left\{\mathbf{x}_t^{(n)}, w_t^{(n)}\right\}_{n=1}^{N}$, which yield the estimate of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

The visualization of the bootstrap algorithm is presented in the Figure[1] 3.1. Note that in the propagating step/ diffusion step, the proposal distribution used in our experiment is a

[1]The figure and notation from the original book [70] are modified for clarification

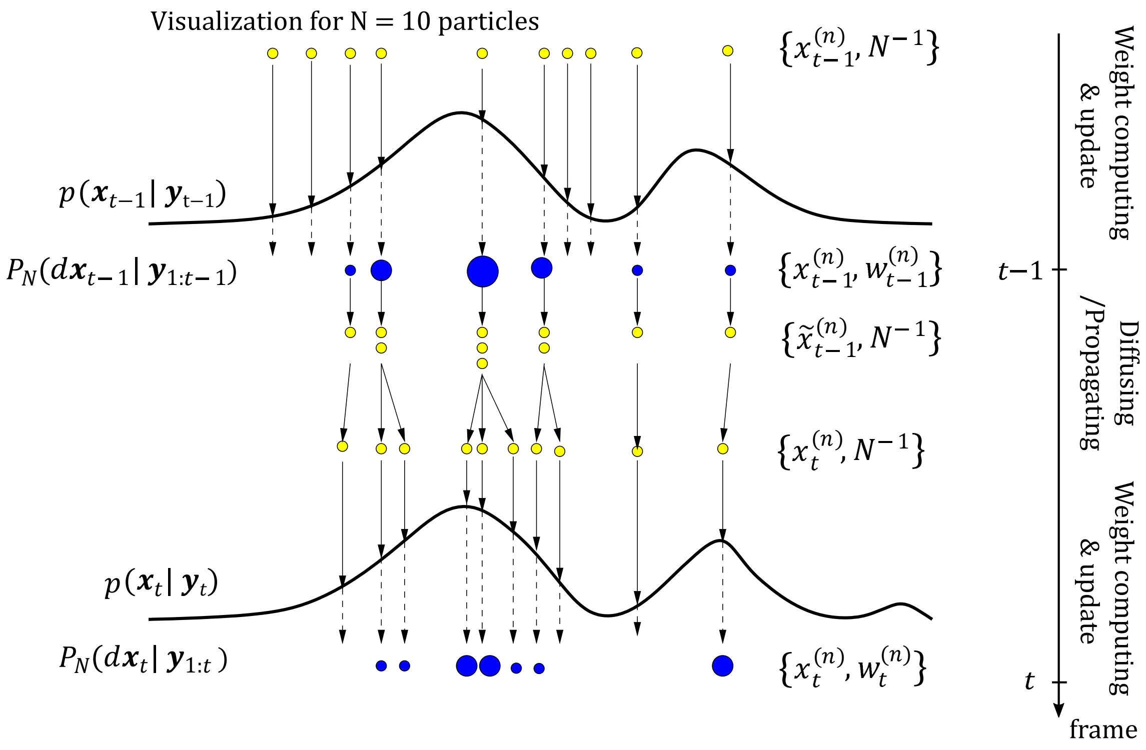

Visualization for N = 10 particles
$\{x_{t-1}^{(n)}, N^{-1}\}$
$p(\boldsymbol{x}_{t-1}|\,\boldsymbol{y}_{t-1})$
$P_N(d\boldsymbol{x}_{t-1}|\,\boldsymbol{y}_{1:t-1})$
$\{x_{t-1}^{(n)}, w_{t-1}^{(n)}\}$
$\{\tilde{x}_{t-1}^{(n)}, N^{-1}\}$
$\{x_t^{(n)}, N^{-1}\}$
$p(\boldsymbol{x}_t|\,\boldsymbol{y}_t)$
$P_N(d\boldsymbol{x}_t|\,\boldsymbol{y}_{1:t})$
$\{x_t^{(n)}, w_t^{(n)}\}$
Weight computing & update
Diffusing /Propagating
Weight computing & update
$t{-}1$
t
frame

Figure 3.1 – In this example, the bootstrap filter starts at time $t-1$ with an unweighted measure $\left\{x_{t-1}^{(n)}, N^{-1}\right\}$, which provides an approximation of $p(x_{t-1}|y_{1:t-2})$. For each particle, its weight is computed from $p(\mathbf{x}_{t-1}|\mathbf{y}_{t-1})$ at time $t-1$. This results in the weighted measure $\left\{x_{t-1}^{(n)}, w_{t-1}^{(n)}\right\}$, which yields an approximation $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$. Subsequently, a new set of unweighted particles $\left\{x_t^{(n)}, N^{-1}\right\}$ are resampled from $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$. Those unweighted particles are still an approximation of $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$. Finally, we weight the particles to obtain the estimate of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ [70].

normal distribution

$$\tilde{x}_t^{(n)} \sim q\left(x_t|x_{t-1}^{(m)}, y_t\right) = \mathcal{N}(\mu = x_{t-1}^{(m)}, \sigma^2). \tag{3.12}$$

The particle representation and the observation $\mathbf{y}_t$ are detailed in the next Section 3.2.

3.2 Sparse coding in visual representation

In this section, we discuss how sparse representation gets involved in the particle filtering framework and how the observation likelihood is defined. In each frame t, the particle filter generates a new set of particles. Each particle represents a possibility of the target object, and it is assigned to an observation, which is called a *target candidate*. All target candidates are denoted as $\left\{\mathbf{y}_t^{(n)}\right\}_{n=1}^{N}$. For convenience, we drop the subscript t out of notation. In the generative approach for object tracking, an appearance model allows the algorithm to select the best candidate among all candidates from the set of particles. Mei et al. [154] first introduced the sparse representation within the particle filter framework. The generative approach models the appearance of the target object as a *sparse linear combination* of a set of templates. In detail, the object appearance is an image patch extracted from the bounding box containing the target and all target *candidates* are represented in the same manner. This yields $\mathbf{y}_t^{(n)} \in R^d$ where d is the size of unrolled image patch vector. Therefore, the particles contain the information of the bounding boxes such as position, size, rotation angle. Each target candidate is approximated by a multiplication of a coefficient vector $\mathbf{a} \in R^p$, whose elements are called *atoms*, and a redundance matrix, called *dictionary*, whose columns are the unrolled image vectors of the target's templates $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K] \in R^{d \times p} (d \ll p)$, such that the coefficient vector $\mathbf{a}$ only contains a few non-zero elements:

$$\mathbf{y}^{(n)} \approx \mathbf{D}\mathbf{a}^{(n)} \qquad \text{s.t. } \ \|a\|_0 \ \text{ is small w.r.t the vector size } \ d, \tag{3.13}$$

where $\|.\|_0$ is the 0-norm of a vector, which is literally the non-zero element-counting operator of the vector. Finding the sparse coefficient vector of a signal in the dictionary is also called encoding the signal in the dictionary. Intuitively, a well-encoded signal has a few atoms in its coefficient vector, it means that the signal has been well-matched with the *words* (i.e., templates) of the dictionary. In the tracking context, a well-encoded candidate means the candidate has been found in the *template* dictionary, as well as the reliability of choosing it as the target. The sparse solution $\mathbf{a}$ of the problem (3.13) is calculated by minimizing the reconstruction error under the sparsity constraint:

$$\mathbf{a}_* = \arg\min_{\mathbf{a}} \|\mathbf{D}\mathbf{a} - \mathbf{y}^{(n)}\|_2^2 \ \text{ s.t. } \ \|\mathbf{a}\|_0 \leq m, \tag{3.14}$$

In the papers of Mei et al. [154, 155], the authors added the trivial templates (*positives* and *negatives*) $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_d, -\mathbf{e}_1, \ldots, -\mathbf{e}_d]$ where each $\mathbf{e}_i$ is a Dirac impulse at location i on a d-length vector, then the dictionary $\mathbf{D}$ is rewritten as:

$$\begin{aligned} \mathbf{D} &= [\mathbf{T}, \mathbf{E}] && (3.15)\\ \text{where} \quad \mathbf{T} &= [\mathbf{t}_1, \ldots, \mathbf{t}_p], && (3.16)\\ \mathbf{E} &= [\mathbf{e}_1, \ldots, \mathbf{e}_d, -\mathbf{e}_1, \ldots, -\mathbf{e}_d]. && (3.17) \end{aligned}$$

During the tracking process, the targets' appearance often changes due to lighting, background, occlusion, or themselves. Although the target's images are well stored in the dictionary, the encoded-vector $\mathbf{a}$ is not really sparse; in other words, it is not well represented. This is caused by a number of pixels on the area bounding the target, as illustrated in Figure 3.2. To compensate those minor changes on sparse representation, the authors added the trivial templates with the number of templates equally large as the number of pixels of the image patch. Each trivial template compensates for only one pixel on the image patch. Therefore, with a few pixels changing their intensity for the mentioned reasons, the encoded vector a could still be sparse. Adding both positive and negative trivial templates makes the encoded vector a contain only the non-negative elements. The sparse representation model with trivial templates of Mei et al. [154, 155] is depicted in the Figure 2.8. In addition to dealing with the changing of some pixels, solving a general ℓ_0-norm optimization problem like Eq. (3.14) is also a hard problem [205, 83]. However,

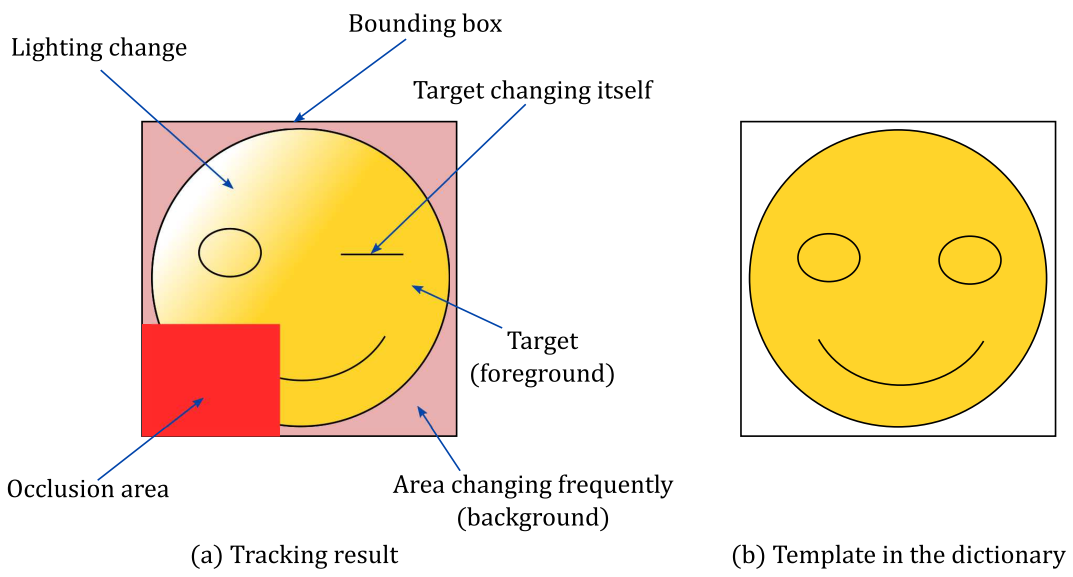


Figure 3.2 – Illustration of the possible appearance changing in tracking videos

its optimal solution can be approximated by resolving its ℓ_1-norm form. Those sparse linear coefficients are computed as the solution of an ℓ_1-penalized non-negative least squares problem:

$$\mathbf{a}_*^{(n)} = \arg\min_{\mathbf{a}\geq 0} \|\mathbf{a}\|_1 + \lambda\|\mathbf{D}\mathbf{a} - \mathbf{y}^{(n)}\|_2^2, \tag{3.18}$$

for fixed $\lambda > 0$. In our implementation, we use the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [16] to solve Eq. (3.18).

For each candidate, we compute an error of reconstruction which is defined as an objective function in Eq. (3.20). These errors yield the weight of particles in order to empirically approximate the likelihood distribution $p(\mathbf{y}_t|\mathbf{x}_t)$ mentioned in Eq. (3.5) and (3.11).

$$p(\mathbf{y}_t|\mathbf{x}_t) \propto \|\mathbf{T}\mathbf{a}_* - \mathbf{y}\|_2. \tag{3.19}$$

The target's estimate is the candidate with the least error among them, that is $\mathbf{y}^* = \mathbf{y}_{i^*}$, where

$$i^* = \arg\min_{1\leq n\leq N} \|\mathbf{T}\mathbf{a}_*^{(n)} - \mathbf{y}^{(n)}\|_2. \tag{3.20}$$

The set of templates $\mathbf{T}$ is updated regularly to adapt to illumination or pose changes and keep the tracker from drifting off the actual target. Moreover, after choosing the best candidate for target, we measure a score to detect the occlusion of targets, called the *sparsity level*, defined as follows:

$$s_{t,k} = \frac{\#\text{activated atoms of } a^{(n*)}}{\text{Size of the dictionary } D} \tag{3.21}$$

Finally, the importance weights $w_t^{(n)}$ are re-computed for each particle and then, the re-sampling step regenerates the particles at time $t+1$ based on the weighted measures

$\left\{\mathbf{x}_t^{(n)}, w_t^{(n)}\right\}$. In the literature, most of sparse tracking approaches try to estimate the observation likelihood in many different ways, for example, [155] uses a Gaussian model whose mean is related the reconstruction error. The next section details our approach of collaborative tracking with multi-camera setting.

3.3 Collaborative multi-camera tracking framework

As we mentioned previously, one of the main challenges of object tracking with a single camera is to cope with long-term occlusion or hard occlusion. In a single view, a target disappears while tracking and then reappears again, its trajectory between those two moments can be implicitly inferred, but not explicitly determined. For this reason, we propose to utilize a set of cameras covering the tracking area from various angles to address this issue, as well as gain reliability and accuracy. This section shows our multi-camera approach to incorporate the tracking results of single cameras in order to improve further the robustness of tracking a single object. We assume that the different cameras have overlapping fields of view and that they are frame-synchronized and calibrated. While we do not precisely specify the hardware capabilities needed to implement our method, we assume that each camera has enough computing power to implement a sparse tracking algorithm. As described earlier, the camera tracks targets on its own video stream, and then compute the 3-D coordinates of targets by using calibration data. Finally, each pair of cameras can exchange lightweight information in the form of 3-D coordinates of a set of points.

Our method exploits the natural observation that target appearance and occlusions generally affect only a dynamic subset of the available cameras. This is illustrated on a synthetic video in the first row of Fig. 3.3 where we present four different views of the same scene. The target object is occluded in the view associated with Camera 2 but visible in all the other views. The crucial point is that, in our algorithm, all these views collaborate to estimate the position of the target *in the frame captured by Camera 2*. This is accomplished by projecting the candidates of Camera 2 into Cameras 1, 3 and 4, using calibration data and then sparsely coding each projected candidate in the local dictionary of each Camera $i \in \{1, 3, 4\}$. The center point of these candidates and their projections are shown in the second row of Fig. 3.3. For convenience, we use the term same "particles" for point clouds, which are the two elements of the state vector of particles $\{x_t, y_t\}$ mentioned in the previous Section 3.1.

We use the reconstruction errors of the local particles and the overall sparsity level of the representation in order to detect the local scene and object variations, and in particular, occlusions. The next subsections detail the different points of implementation to solve different practical issues. We begin with a critical issue while using the particles from a single view to estimate the empirical distribution for another view.

3.3.1 Common ground-plane particle filter

This section describes a critical issue of using the local particle filter in a collaborative tracking approach with multi-cameras, which is mentioned at the beginning of Section 3.3,

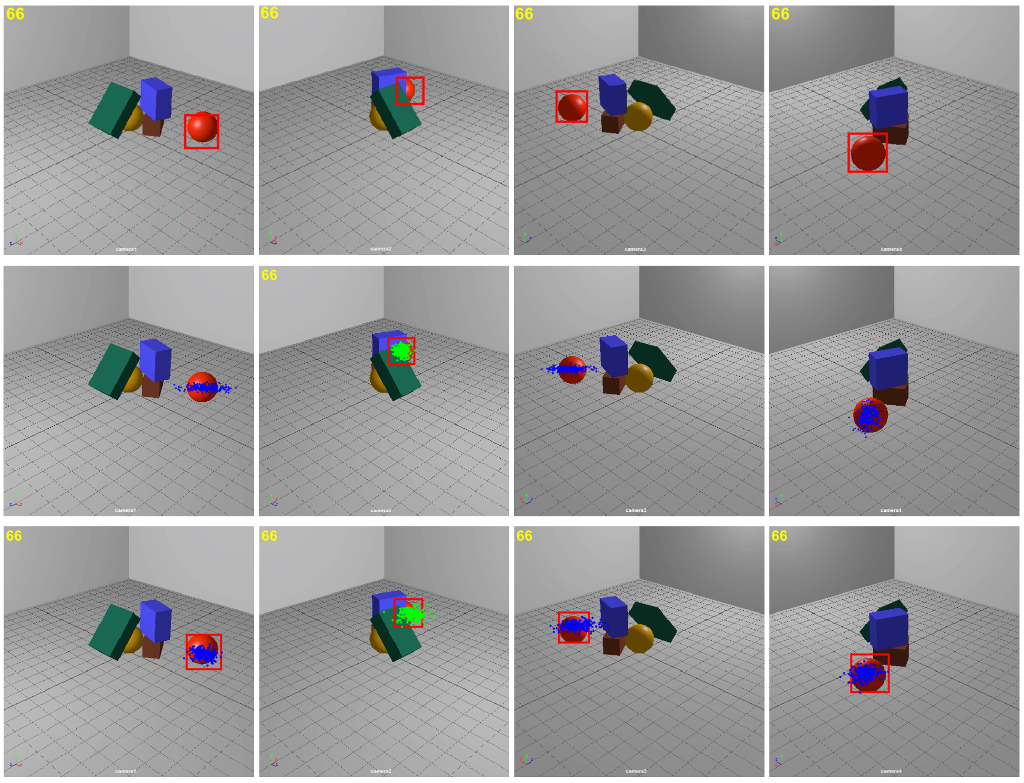


Figure 3.3 – Illustration of the multi-camera collaborative tracking approach. From left to right: frame 66 of Camera 1 (F66C1) (no occlusion), F66C2 (occlusion), F66C3 (no occlusion), F66C4 (no occlusion); First row: the particles in view 2 (green) are projected into other view (blue). Second row: the projected particles on the ground plane into each single view. Third row: same as second row but particles are spreading directly on the common ground.

as well as our proposed approach. In practice, projecting the particles from one camera into another makes the distribution estimate on that camera change, which means that the distribution of particles no longer represents the initial empirical distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ on neither the initial view or the perspective view. The distribution on the image plane (i.e., 2D space) in each camera is not the distribution of the "presence" probability of the target on the 3D space and the other views either. An illustration of this problem is provided in the second row of the Figure 3.3 where the particles (i.e., the center points of the candidates) in Camera 2 are projected into the other ones. The distribution of the particles projected into Camera 4 quite resembles those in the initial view (Camera 2), while those on the other's view (Camera 1 and 3) is spreading and oriented toward the direction of Camera 2. In this illustration, when the local point cloud of candidates in a camera is slightly displaced, then its corresponding projections into other cameras may stand out of the target. Unquestionably, the angles made by camera-object-camera and camera-object-plane directly affect the distribution of particles between the different views, and this local particle projection into other views ineffectively approximates the distribution of the target.

To overcome this serious problem, we propose a novel particle filtering setting in which we change the coordinates of the particles from *pixels* in the image plane to meters in the $z = 0$ plane (the ground plane) of the 3-D world. In other words, instead of using the particle filter on the image plane, we deploy them directly to the plane $z = 0$ of the real world. The third row of Fig. 3.3 shows the result of implementing this strategy on a synthetic video. The spread of projected particles (the blue ones) should be contrasted with that obtained in the second row, where a standard *image plane* particle filter is applied. One can indeed see that the spread of the candidates is less affected by a camera change. This does not only facilitate the collaborative process between cameras but also reunites all the variance coordinates of the different cameras into one. In comparison with the *image plane* particle filtering, where one target has different affine parameters in each view, the novel *common ground-plane* particle filter makes all the views share the same parameters. This means that all cameras in the network perform particle filtering with the same parameters in order to track the same target. The changes of particle filters from image planes to a common ground-plane lead to the reformulation of the likelihood distribution (3.19) on each view.

$$\begin{aligned} p(\mathbf{y}_{t,k}|\mathbf{x}_{t,k}) = p\left(G_k(\mathbf{y}_t)|H_k(\mathbf{x}_t)\right) &= p_k(\mathbf{y}_t|\mathbf{x}_t) \propto \|\mathbf{T}_k\mathbf{a}_* - \mathbf{y}_k\|_2 \\ \text{where :} \qquad k &= \text{the camera number} \\ G_k(\mathbf{y}) &= \mathbf{y}_k \\ H_k &= \text{the homography function converting the points} \\ & \text{ on image-plane to ground-plane of Camera } k \end{aligned} \tag{3.22}$$

On each view, the likelihood distribution $p_k(\mathbf{y}_t|\mathbf{x}_t)$ are defined as Eq. (3.22). Notice that $\mathbf{y}$ is the multi-view observation vector consisting of all observations on single view $\mathbf{y}_k$. The visualization of common ground-plane particle filter performing on the dataset PETS09-S2L1 is presented in the Figure 3.4.

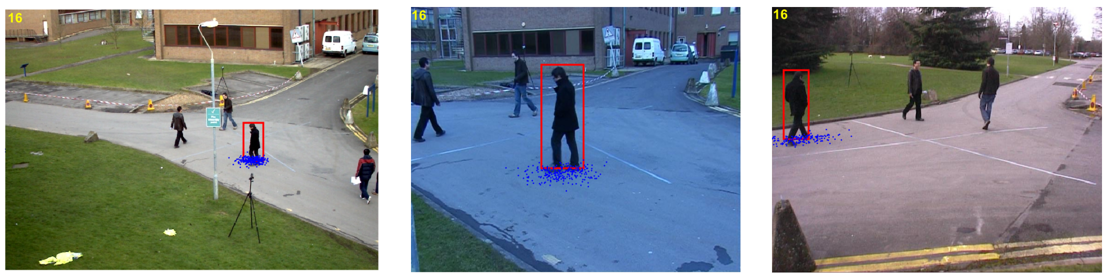

(a) Camera 1 (b) Camera 6 (c) Camera 7

Figure 3.4 – Visualization of the common ground-plane particle filter in multi-camera tracking

3.3.2 Template update strategy

One of the main problems of visual tracking, which is set to be explained in this section, is the over-updating of the appearance model. In reality, target objects usually change during tracking videos due to self-variation (i.e., non-rigid object), lighting condition, pose changing. Therefore, regularly updating the target's appearance is necessary to maintain a good track. However, updating the appearance model every frame can result in a severe issue, called *drift problem*, which is observed when the appearance templates are gradually shifted or *drifted* out of the actual target after a long time of tracking. An example of the drift problem is illustrated in the Figure 3.5.

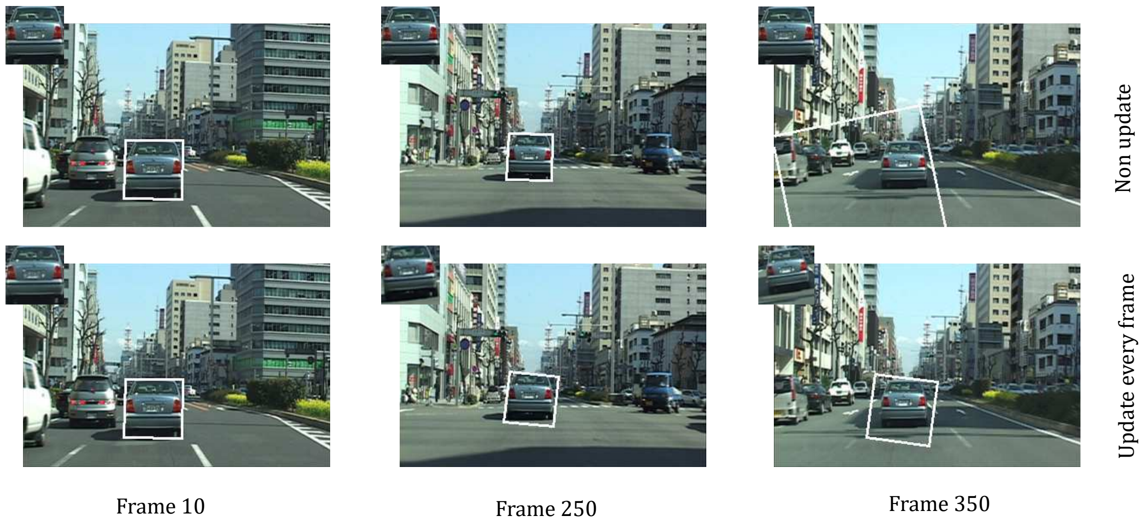


Figure 3.5 – An example of drift problem in [153]. The first row illustrates the tracker is failing to track the car after not updating any new target's appearance at all. The second row shows the templates being updated every frame which leads to the undesirable updates resulting in drift problem.

In our framework, the issue of updating the templates dictionary becomes worse, especially for non-rigid object tracking. To address this typical issue, we propose several techniques while updating the target's dictionary. Firstly, we consider that the target is being occluded if the sparsity level s_t of the chosen candidate is greater than a predefined threshold ϵ_1. Otherwise, the candidate is the target's estimate for tracking results. As discussed in Sec. 3.2, smaller the sparsity level is, the more the current candidate looks like the initial templates in the dictionary. For an accurate update, s_t needs to be much smaller than ϵ_1 and we choose another threshold $\epsilon_2 < \epsilon_1$ to put the candidate in the dictionary (if $s_t < \epsilon_2$). If the sparsity level s_t stands in between ϵ_2 and ϵ_1, the target is considered as being partly occluded or having a brutal appearance variation. In this case, the dictionary does not update any new template.

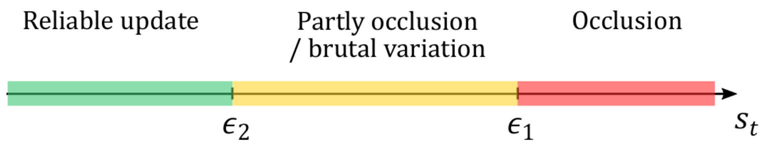


Figure 3.6 – The different sparsity levels to determine if the target is occluded or reliable to update. When $s_t < \epsilon_2$, it is safe to update the target template into the dictionary, otherwise, it is too ricky to update. When the sparsity level s_t exceeds ϵ_1 value, that triggers the collaborative tracking process between cameras.

Secondly, as an effort to keep the updates precise, we lighten the update requirement by adding a condition about the disparity of particle distribution on the ground plane between cameras, which is the Bhattacharyya distance between the particle clouds in different cameras on the ground-plane. This distance is supposed to be smaller than a separative threshold distance ρ to confirm the position of the target. This means that there is no drift problem detected on any view because the possibility of the trackers from all views being drifted toward a direction at the same time is relatively insignificant. In detail, when a camera wants to update the appearance templates of its target, it performs a confirmatory check by measuring the distance between its estimated distribution (the set of particles) and those from the other cameras tracking the same target. Therefore, the distance between two distributions from two cameras C_k and C_l are measured by the Bhattacharyya formula:

$$d(C_k, C_l) = D_{Bhattacharyya}\left(p_k(\mathbf{x}_t|\mathbf{y}_t), p_l(\mathbf{x}_t|\mathbf{y}_t)\right) \tag{3.23}$$

If those distances are all smaller than a predefined threshold distance ρ, the target is available to update, otherwise, it is not. The separative thresholding is illustrated in the Figure 3.7.

The additional condition for updating the target template on a single camera C^k, $f(k)$, is reformulated as follows:

$$f(k) = \begin{cases} 1, & \text{if } \forall i \neq k, d\left(C_k, C_i\right) < \rho. \\ 0, & \text{otherwise.} \end{cases} \tag{3.24}$$

Finally, we set up an update timer m_k as well to prevent our algorithm from over-updating. This counter is incremented after each frame, and the update process does not occur unless the counter exceeds a minimum update time τ. The counter is reset once the update is operated. More details can be seen in Alg. 3. The next section discusses

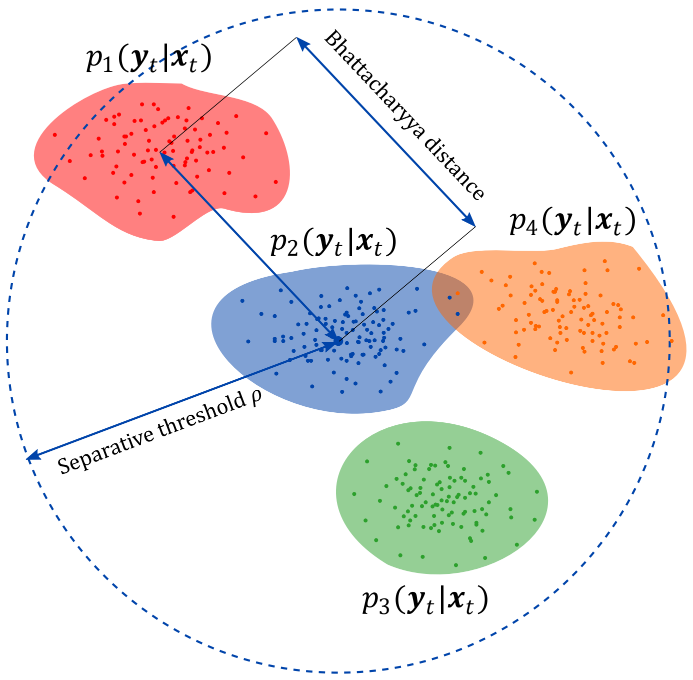

$p_1(\boldsymbol{y}_t|\boldsymbol{x}_t)$
Bhattacharyya distance
$p_2(\boldsymbol{y}_t|\boldsymbol{x}_t)$
$p_4(\boldsymbol{y}_t|\boldsymbol{x}_t)$
Separative threshold ρ
$p_3(\boldsymbol{y}_t|\boldsymbol{x}_t)$

Figure 3.7 – The separative thresholding to confirm the target's position to avoid drift problem.

the architecture of our multi-camera setting, which prevents errors from propagating in the camera network, as well as, allows multiple cameras to collaborate effectively to track individuals moving between different FOV of cameras inside the network.

3.3.3 Network architecture for recovering tracking state and propagating error prevention

This section details the camera network architecture of our multi-camera tracking framework. The architecture promotes cameras to communicate with each other their own tracking results for their collaboration and to recover the tracking process of the cameras whose targets are lost while preventing the false tracking result from propagating inside of the network.

In practice, there are significant problems when implementing the tracking algorithm in multiple cameras. Firstly, since a target is not necessarily present in all the FOV of cameras all the time, it might appear and disappear in certain cameras. In other words, when a target moves around the tracking area, it is often the case where only a subset of cameras can see it, and this camera subset changes following the target's position. To address this issue, we develop a feature to manage all the exits/entries for each camera. Concretely, our algorithm pauses the tracker of a particular camera if the target leaves its FOV. Secondly, when the target returns into its FOV, our algorithm resumes the tracker and informs the new position of the target. To support this algorithm, we set up our camera network as a star network with a center camera and its neighbor cameras. The main camera has the largest view, among others. It brings a privilege for the collaborative tracking fashion described previously for several reasons. First, the main camera observes most of the scene, and it is favorable to acknowledge and confirm to others when a target is entering into their FOV or whether it is the same target which used to be tracked in previous frames. Secondly, in the case of the main camera losses a target, e.g., occlusion, the other cameras help it to localize the target during the occlusion. Figure 3.8 illustrates the communication between cameras inside a star network model.

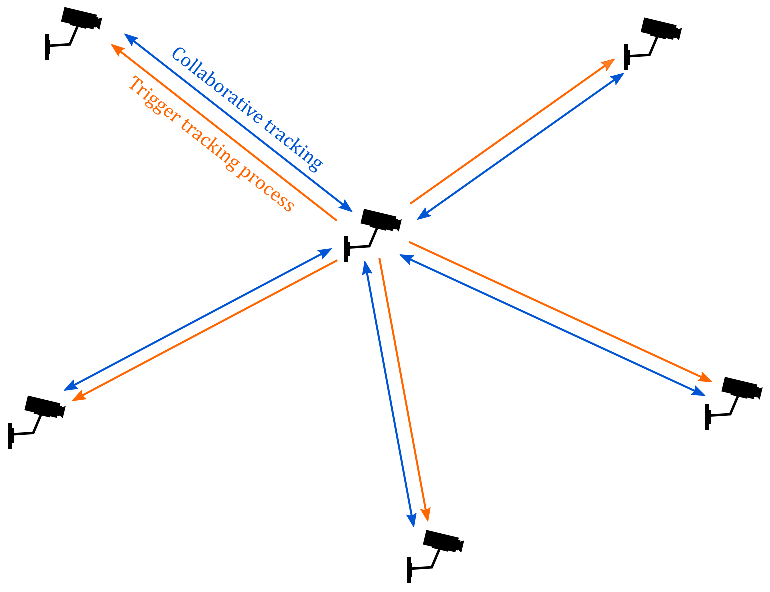


Figure 3.8 – The star topology for collaborative multi-camera network.

Furthermore, this architecture eliminates the early tracking error from a single camera which potentially spreads in the network, because the communication is established between the center node with its peripheral cameras. All collaboration must go through the center camera, so in the case, the error contaminated the center node, this tracking result will quickly be removed while comparing it with other peripheral cameras whose results are more reliable. Finally, the star topology leads to a strong computational benefit comparing with a complete graph topology. In turn, when the master camera loses the target, it recovers its state from other cameras. In detail, as a camera recognizes that another one is missing the target, it projects all its particles to the other one, which will encode these candidates. If the target were found among them, the tracking-state in this view would be reestablished. This whole star topology could prevent the false tracking result of one camera from contaminating other ones. The detailed steps are summarized in Algorithm 2. In the next section, we will detail the collaborative tracking procedure inside our multi-camera setting.

```
Input  : $K$ cameras $C_1, \dots, C_K$, $\epsilon_1$ and $\epsilon_2$ occlusion sparsity thresholds, $\lambda$ in
         Eq.(3.18), $\tau$: minimum update time, $\rho$: separative threshold distance
Output: Trajectory of the target in each camera
1  Initialization of each camera;
2  Main loop:
3  foreach frame in videos do
4      foreach camera $C_k$ do
5          if Target appears in observable zone then $\mathbf{y}_{t,k}^{(n^*)}$, $s_{t,k}$ ←Auto-tracking in
             Sec. 3.2;
6          else break;
7          if $s_{t,k} < \epsilon_2$ then
8              Set non-occlusion state for $C_k$;
9              Select the candidate $\mathbf{y}_{t,k}^{(n^*)}$ as the target;
10         else
11             Set occlusion state for $C_k$;
12             Project the particles of $C_k$ into $C_{k'}, k' \neq k$, get the candidates $\mathbf{y}_{t,k'}^{(n)}$;
13             Sparse coding $\mathbf{y}_{t,k'}^{(n)}$ by Eq.(3.18) on $C_{k'}$ 's dictionary ;
14             Select $\mathbf{y}_{t,k'}^{(n^*)}$ as Eq.(3.20);
15         end
16         $\gamma \leftarrow$ Bhattacharyya distance of the particles of $C_k$ and all those of others
             cameras;
17         if $\gamma < \rho$ & $s_{t,k} < \epsilon_2$ & $m_k \geq \tau$ then Update the dictionary and $m_k \leftarrow 0$ ;
18         else $m_k \leftarrow m_k + 1$;
19         if Target missing then Get target's position from $C_{k'}, k' \neq k$ by Algo. (2);
20     end
21 end
```
Algorithm 1: Single-object Online Multi-camera Collaborative Tracking

```
  Input  : $C_k$ successfully tracks the target, $C'_k$ whose target is missing, $t$: frame, $\epsilon$
           occlusion sparsity threshold, $\lambda$ in Eq.(3.18)
  Output: $C_{k'}$ with the state-tracking recovered
1 if Target in FOV of $C_{k'}$ then
2     Project the particles $x^{(n)}$ of $C_k$ into the image plane of $C_{k'}$, then get the
        candidates $\mathbf{y}_{t,k'}^{(n)}, 1 \leq n \leq N$;
3     Sparse coding for each $\mathbf{y}_{t,k'}^{(n)}$ as Eq.(3.18) ;
4     Compute the reconstruction error for each $\mathbf{y}_{t,k'}^{(n)}$ with the templates in its
        dictionary;
5     Select the candidate $\mathbf{y}_{t,k'}^{(n^*)}$ as Eq.(3.20);
6     Compute the sparsity level $s_{t,k'}$;
7     if $s_{t,k'} < \epsilon$ then Accept the recovery;
8     else Reject the recovery;
9 end
```

Algorithm 2: Recovery of tracking state

3.4 The proposed algorithm

In this section, we concretely describe our collaborative tracking approach in the multi-camera setting. Given a set of cameras $\{C_k\}_{k=1}^K$, each camera first projects the particles of the target on its image plane and sparsely encodes the candidates extracted from these particles $\mathbf{y}_{t,k}^{(n)}$ with its local dictionary $\mathbf{D}_k$ as we described in Sec. 3.2. Next, the weight of each particle is assigned with the reconstruction error (Eq. (3.19)) of its corresponding candidate as the weight update step as Eq. (3.11). Those weighted particles $\left\{\mathbf{x}_{t,k}^{(n)}, w_{t,l}^{(n)}\right\}_{n=1}^N$ yields the empirical likelihood distribution $\left\{p_k^{(n)}(\mathbf{y}_t|\mathbf{x}_t)\right\}_{n=1}^N$, or $\{p^{(n)}(\mathbf{y}_{t,k}|\mathbf{x}_{t,k})\}_{n=1}^N$. In the case of failure (e.g. occlusion detected or target missing) which is signaled by the sparsity level s_t exceeding the limit threshold ϵ_1, the current camera C_k demands the cameras $C_{k'}$, $k' \neq k$, whose target is not occluded, to process a substitute encoding of the particles on C_k. Notice that during this "backup" sparse coding on $C_{k'}$, the coordinates of particles are taken from the initial camera C_k, and the candidates are built up with the parameters and image frame from $C_{k'}$. In other words, the local empirical likelihood estimate on C_k is replaced by an alternative empirical likelihood estimate done by $C_{k'}$:

$$\left\{p_k^{(n)}(\mathbf{y}_t|\mathbf{x}_t)\right\}_{n=1}^N \approx \left\{p^{(n)}(\mathbf{y}_{t,k'}|\mathbf{x}_{t,k})\right\}_{n=1}^N \tag{3.25}$$

It is worth noting that the parameters needed to compute a bounding box in each camera $C_{k'}$ are not sent by C_k, but the algorithm uses each camera's current own scale and aspect ratio parameters. Finally, having the substitute likelihood estimate from $C_{k'}$, the particle with the least reconstruction error is chosen as the position of the target on Camera C_k. Eventually, the substitute distribution replaces the old likelihood distribution on Camera C_k, which is needed for the diffusion step in the next frame. The entire algorithm is shown in Alg. 3.

In our framework, we also develop a feature to recover the tracking state of the cameras whose targets are being missing by tracking failure or just entering their FOV. In reality, a camera might lose its target during tracking process by many reasons, being able to recover its tracking process of every single camera makes the multi-camera network more robust and accurate in the context where a target can move in or out of FOV's cameras multiple times during tracking videos. The general idea is that any camera can be signaled to resume its tracking process at any time after losing its target by another camera. Indeed, in every frame, the algorithm checks if, for every single camera, an individual being tracked by other cameras is present in its FOV. If it is the case, the camera receives the set of particles from one other camera and starts encoding the candidates extracted from those particles with its exiting dictionary D_k. In the case when the best candidate found is well represented via its low sparsity level, this resumes the tracking process on the camera C_k. The detail of the recovery procedure is shown in Algorithm 2. The next section will reveal the experiments conducted on a Multi-Camera dataset, which validates the efficiency of our multi-camera approach comparing with single-camera ones.

3.5 Experimental results

In this section, we present the experimental results obtained from our collaborative multi-camera approach to track mono-object. In the literature, numerous benchmark datasets have been developed for generic object tracking [7, 130, 181]. However, most of them are not dedicated for multiple view tracking problems, because they are realized in a single view, or some of them used the cameras moving and being carried by a person which are not for surveillance purpose that we seek for. Some datasets do not provide any calibration information of cameras such as the USC Campus [129]. Otherwise, DukeMTMC datasets [178] is large, and its overlapped zones are relatively small and with the purpose of a re-identification benchmark. Fortunately, with multiple calibrated, synchronized cameras having an overlapped zone, the PETS2009 [78] dataset is the best option for our purpose. For our experiments, we use the sequence “S2L1” which are noticed for tracking purpose by the authors [78]. The sequence “S2L2” is unfitting because of the limit of the two synchronized views and the density of crowds. For assessing the performance of our method, we set up a variety of multi-view configurations, including three views (158, 168 and 178), four views (1678), and six views (135678). We choose to track the person with ID 1, which has the most complex trajectory in the video Fig. 3.9, but the other targets could have the same evaluation. For the evaluation metric, we use the average overlap score (AOS) [74], a commonly used metric for object tracking evaluation [228]. The AOS score is straightforwardly the average of the Intersection over Union (IoU) score on the entire image sequence.

In order to prove the efficiency of our approach, we compare our multi-view method with the single view version, which is originally the L1 tracker [155] with the particle filtering on the ground plane as described in Sec. 3.1. We also note that the sparse tracking algorithms such as [250] [155], [112], [95] can leverage our framework to work in a multiple cameras setting. We followed this line by extending Sparsity-based Collaborative Model (**SCM**) [250]. The results in Tab. 3.1 clearly demonstrate the effectiveness of using

Methods	View	Single view	Multiple view				
			157	168	178	1678	135678
L1[155]	1	0.553	-	0.630	**0.636**	0.573	0.612
	5	0.433	0.575	-	-	-	**0.611**
	8	0.340	-	0.566	**0.616**	0.604	0.611
SCM[250]	1	0.489	-	**0.544**	0.508	0.466	0.521
	5	0.394	0.517	-	-	-	**0.566**
	8	0.479	-	0.630	**0.636**	0.573	0.612

Table 3.1 – Comparison of single-view methods with different multiple view configurations. The **bold** value represents the best result on each *line*.

multiple views to help a single view to track the identity while being hidden. Significant improvements are shown in both baseline single-view methods L1 and SCM. The score increases from about 2% to 27.1%, depending on configurations.

Alternatively, we compare our method to several algorithms in the literature including Multiple Instance Learning (**MIL**) [7], Tracking Learning Detection (**TLD**) [116], Kernelized Correlation Filters (**KCF**) [102], **STRUCK** [95] and Adaptive Structural Local Sparse Appearance Model (**ASLA**) [112]. The results are shown in Tab. 3.2 with the median values of the multi-view approaches over different configurations in the last rows. The superiority of our approach can be seen while comparing to the single-view tracking algorithms in the state-of-the-art.

Methods	view 1	view 5	view 8
MIL[7]	0.159	0.315	0.100
TLD[116]	0.171	0.182	0.119
BOOSTING [90]	0.039	0.260	0.175
STRUCK[95]	0.321	0.282	0.175
KCF[102]	0.138	0.018	0.013
ASLA[112]	0.478	0.541	0.506
SCM[250]	0.498	0.394	0.479
Baseline-L1[155]	***0.553***	0.433	0.340
SCM Multi-view	0.508	***0.542***	***0.578***
L1 Multi-view	**0.630**	**0.593**	**0.608**

Table 3.2 – Comparison of State-of-the-art methods to our multi-view method on the "PETS09-S2L1"

To qualitatively evaluate the effectiveness of our approach, Fig. 3.9 shows the results of TLD [116] and our method with the configuration *cam168*. As a result of tracking with a single view, TLD could not track the identity while being hidden by the road sign, but with two collaborative views, the proposed algorithm had this capacity instead.

To end our experimentation, we investigate the impact of parameters on multi-camera tracking process. As mentioned above, there are many parameters to adjust in different steps of tracking process including variance σ in particle propagation (3.12) in particle filter, the thresholds ϵ_1, ϵ_2 (3.6) determine occlusion levels of targets, or separative threshold ρ (3.7). However, the factor having the most direct impact on the collaboration between cameras when occlusion occurs is the proposal distribution, which defines where to find target in the next frame. In addition, the dispersion of the particles around a target effects the ability of being processed by neighbor cameras, because if no camera finds any close-

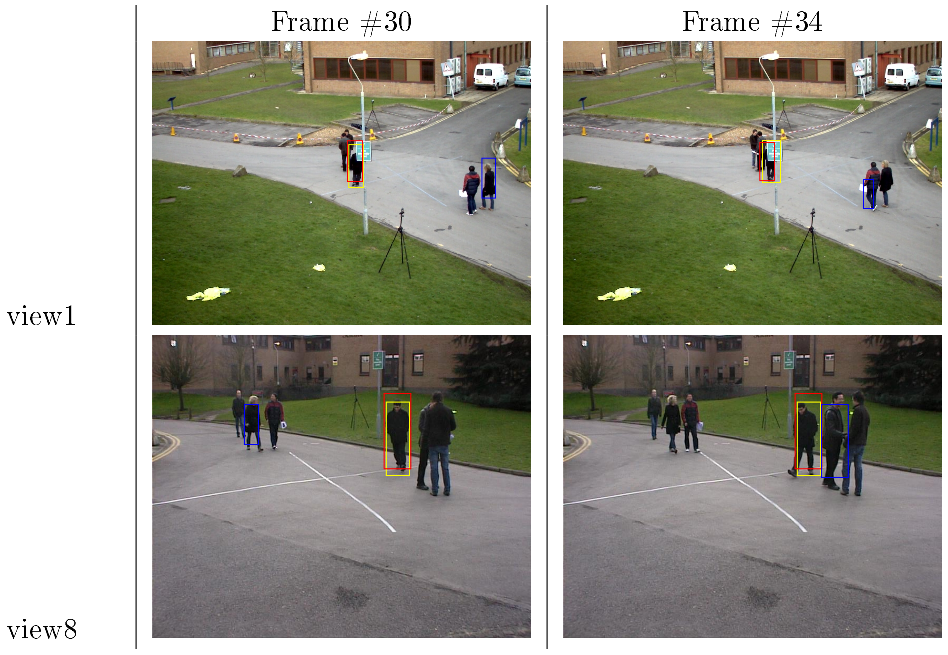


Figure 3.9 – Visual results with the yellow, blue and red boxes are respectively the ground-truth, TLD [116] and our multi-view method.

by particle "cloud" (via a separative threshold ρ), the collaborative tracking will not be applied. Hence, the variance $\sigma = [\sigma_x, \sigma_y, \sigma_h, \sigma_w]$ appears to be most impactful in this the study. In this experiment, we varied only position variance σ_x, σ_y from 120 to 220, the others remain constant.

As the final objective of tracking is applying in general multiple object tracking case, we deployed our approach on all targets on the sequence and use MOT-CLEAR [22] to evaluate the performance 2 main scores MOTA-Multiple Object Tracking Accuracy and MOTP-Multiple Object Tracking Precision. We tried implementing our sparse coding tracking approach on the MOT context, but due to the mis-managing trackers when multiple targets interacting during their movement, the initial approach could not produce desired outcome. Therefore, we replaced sparse-coding-based part by a more determinist approach (KCF-Kernel Correlation Filter [102]) meanwhile keeping the proposal particle propagation. We also recorded the collaboration times between camera pairs. The result table Tab. 3.3 shows the optimal value of σ_{xy} at 150 (for full graph network model) and at 220 (for star graph network model). When we increase the variance σ_{xy}

σ_{xy}	MOTA	MOTP
120	27.241	66.418
120*	28.04	66.723
150	28.753	**66.778**
150*	28.047	66.725
175	26.516	66.346
175*	27.485	66.428
200	26.448	66.321
200*	29.259	66.468
220	24.742	66.232
220*	**29.583**	66.529

Table 3.3 – Impact analysis on different configurations. This is the example of σ_{xy} The **bold** value represents the best results. *: Experiments with the star topology camera network 3.8.

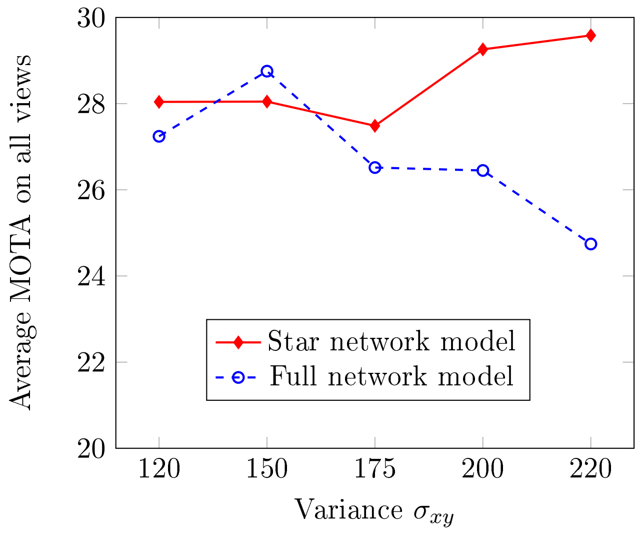
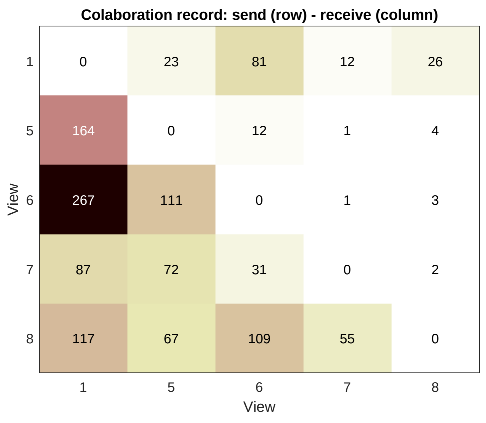
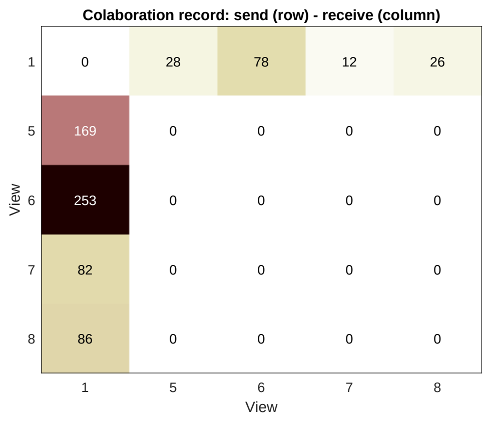


Figure 3.10 – MOTA score in function of variance σ_{xy}. The performance of collaborative star-network camera model shows its efficiency on preventing error propagation across camera network.

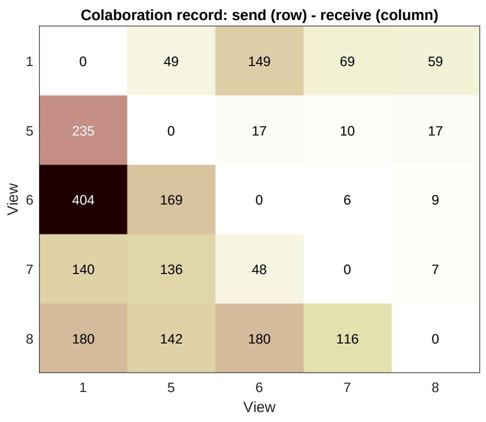
from the optimal value 150 to 220, the tracking per-
formance decreases rapidly, because of the spread of the inaccurate tracking results transferring in the camera network. This can be implied from the collaboration record showed the figures 3.11 a) and c). The number of the demands for collaborative tracking in each camera dramatically rose when the high-value of σ facilitates collaboration. Consequently, the collaboration also promotes the error propagation inside the full-graph network. Meanwhile, within the star graph model for collaboration, even though the value of σ is high, the error is more likely to be contained locally (see Fig. 3.11 b) and d)) . As a result, the camera star-network remains (or even increase) the tracking performance (Tab. 3.3 and Fig. 3.10). Besides the analysis retreated above from the collaboration tables Fig. 3.11, the records also show us that the view 8 and 6 are the most demanding views, as they observe most movements of the actors on the scenes.



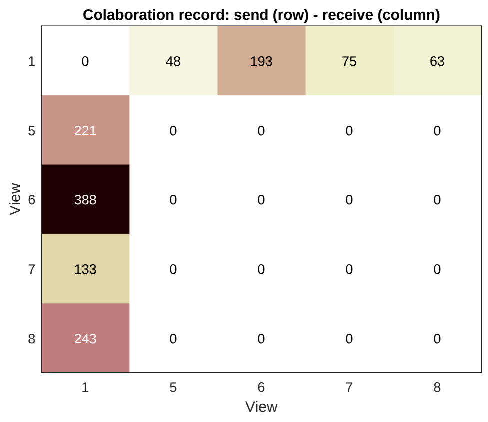
a) $\sigma_{xy} = 150$, full graph network



b) $\sigma_{xy} = 150$, star graph network



c) $\sigma_{xy} = 200$, full graph network



d) $\sigma_{xy} = 200$, star graph network

Figure 3.11 – Collaboration record between cameras in Sequence PETS09-S2L1 with different propagation values σ_{xy}. Rows indicate demands sending to other cameras when occlusion occurs. Columns indicate demands receiving from other cameras. The figures on the left use the full graph model for collaboration between cameras, those on the right use the star model described in Fig. 3.8

Chapter 4

Multiple object tracking: Target association across multiple cameras

This chapter presents our multi-camera framework that allows a single camera to collaborate with other cameras of the network in order to enhance tracking performance. Our approach adopts a Multiple Object Tracking framework for multiple Single-Object-Tracking trackers in a multi-camera setting. This chapter is separated into three sections. The first section describes our multi-camera tracking framework to handle target occlusion, which involves associating targets across different views. Therefore, in the second section, we introduce our first association method to match targets in *all* cameras, which is based on graph-based clustering. The final section presents our second association method to assign targets in pairs of cameras, in which we reformulate the assignment problem as an unbalanced optimal transport. The experimental results show the efficiency of the multiple cameras system for multiple object tracking in comparison with single camera systems. For convenience, we list all notations used in this chapter as follows:

v_m^k	The m^{th} target in the camera k
V	The set of nodes representing targets $\{v_m^k\}$
E	The set of edges connect nodes
$G = (V, E, w)$	The full graph associating targets across all cameras
$C_k = \{v_1^k, ..., v_M^k\}$	The cluster of targets in the camera k
$G_s = (V_s, E_s, w_s)$	The subgraph of the full graph G indicating a specific identity
$V_s = \{v_m^k \vert k \in \{1, ..., K\}\}$	The set of targets across views of an identity
$E_s = \{E(p, q) \vert p, q \in V_s\}$	The set of edges linking all targets in the subgraph G_s
$w_s = \{w(p, q) \vert p, q \in V_s\}$	The set of weights assigning to each edge in G_s
$\mathcal{T} = \{v_j^k\}_{j=1}^N$	The set of tracker representing targets $\{v_m^k\}$
$x_m^{k,i}$	The coordinate of the target v_m^k at frame i
$\mathbf{x}_m^k = \{x_m^{k,1} ..., x_m^{k,F}\}$	The trajectory feature vector of the target v_m^k
Φ_m^k	The bounding box of the target v_m^k at current time
f_{app}	The appearance feature distance function
f_{traj}	The trajectory distance function

α, β	The discrete measures with weights, called respectively as *source* and *target*
δ_x	The Dirac at position x
$\mathbf{a} \in \mathbb{R}_+^n$, $\mathbf{b} \in \mathbb{R}_+^m$	The weights of discrete measures
$\mathbf{P} \in \mathbb{R}_+^{n \times m}$	The optimal transport plan (i.e., coupling matrix) matching from a source set of n elements to a target set of m elements
$\mathbf{C} \in \mathbb{R}_+^{n \times m}$	The distance matrix or pairwise cost matrix
$\mathbf{f} \in \mathbb{R}_+^n$, $\mathbf{g} \in \mathbb{R}_+^m$	The potentials which are dual variables from the dual problem of optimal transport
$\mathbf{R}(\mathbf{C})$	The set of admissible dual variables
$\mathbf{L_C}(\mathbf{a}, \mathbf{b})$	The optimal cost in Kantorovich's problem given a distance matrix $\mathbf{C}$
$\mathbf{H}(\mathbf{P})$	The entropy of a distribution $\mathbf{P}$
ε, τ	The regularization parameters

4.1 General online multi-camera multi-object tracking framework

This section details our multi-camera tracking framework, which mainly focuses on using multiple cameras to tackle the occlusion problem in tracking, as discussed previously. This framework requires multiple cameras in the network that has overlapping fields of view. In order to incorporate the results of every single view, all cameras need to be well-calibrated and synchronized. Having all tracking results from all cameras, an association process matches, across cameras, the trackers that belong to the same identities. Since the association result is available, when one camera loses its target (e.g., occlusion, out of field-of-view), it is possible to trace back its missing targets from the tracking results of other cameras in case of the target reappears again.

4.1.1 Tracker management in online multi-camera multi-object algorithms

In this section, we present the mechanism of organizing multiple SOT trackers to track multiple objects simultaneously in a multi-camera setting. As mentioned in the chapter of the state-of-the-art, one of the most challenging problems of implementing SOT algorithms to perform MOT tasks is to manage and organize all SOT trackers to avoid redundancy, identity switches, as well as, to deploy detections in every frame efficiently.

Adopting the Markov Decision Process (MDP) framework of Xiang et al. [229], any individual in the FOV of each camera is supposed to be tracked by a SOT tracker. As a Markov Decision Process, a tracker possesses 3 possible states: *active*, *inactive*, *tracked* and *lost*. The MDP model of Xiang et al. [229] is illustrated in Figure 4.1. The state of trackers moves through these states which correspond to different situations from the beginning to the end:

- Initialization: When a tracker is created from a detection, it possesses the “active” state.

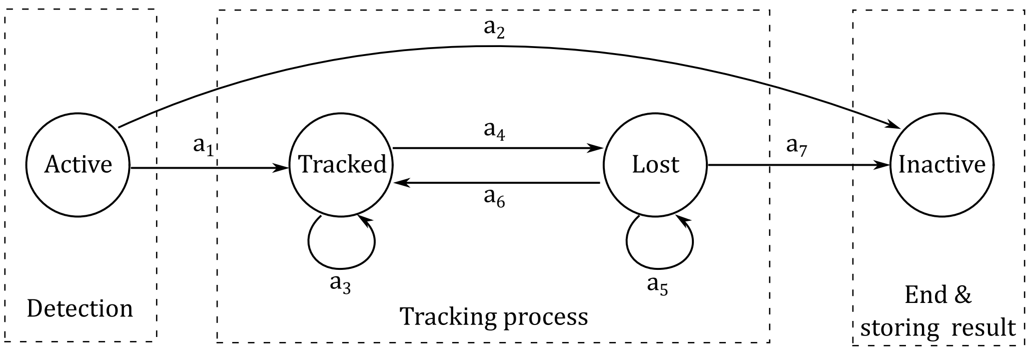


Figure 4.1 – The Markov Decision Process for a single SOT tracker.

- Tracking in process: When a tracker succeeds to track its target at the current frame, it possesses the "tracked" state.
- Tracking pause: When a tracker loses its target at the current frame, it possesses the "lost" state.
- Finished tracking: When a tracker stops to track its target, it possesses the "inactive" state.

The Markov Decision Process is completed by defining the deterministic transitions allowing the tracker to transfer between the possible states:

- If an "active" tracker (i.e., new born tracker) is classified as a false positive, the state of the tracker is transferred from "active" to "inactive" (i.e., the transition a_2), or else to "tracked" (i.e., the transition a_1).
- If a "tracked" tracker succeeds to track its target at the current frame, its state remains as "tracked" state in the next frame (i.e., the transition a_3), otherwise, it transfers its state to "lost" (i.e., the transition a_4).
- If a "lost" tracker recaptures its lost target, it transfers its state to "tracked" (i.e., the transition a_6), otherwise, it remains at its "lost" state (i.e., the transition a_5). In the case of a long time being in "lost" state (e.g. 30 frames, for example), the tracker transfers its state to "inactive" and ends its tracking process (i.e., the transition a_7).
- Any tracker whose state is "inactive" will no longer return to the tracking process.

In principle, the collaborative tracking process within our multi-camera network is separated into two parts: independent tracking on each camera and associating data across cameras. In terms of independent tracking from a single view (i.e., a single camera), the MDPs of the trackers operate similarly to the MOT method of Xiang et al. [229]. Detections are first classified by an SVM to determine whether the new tracker is initialized and transferred to the tracking process (i.e., true positive) or goes directly to "inactive" state (i.e., false positive). When a tracker is in "tracked" state, it operates a SOT algorithm to track its target. In the case of failure, the tracking process is replaced by a data association process, which matches the trackers with a detection belonging to it. This might go through two "back up" steps: the single-camera matching ([229]) and the across camera matching (our contribution). First, the camera itself tries to match the tracker with detections nearby. If this matching succeeds, the second step is not necessary. Otherwise, the data association result from all cameras is used to match the tracker with a detection. After going through 2 matching steps, if the tracker succeeds to match with its detection, it remains in its "tracked" state in the next frame, or else transfers its state to "lost". When the tracker is in "lost" state, it remains its state until the matching algorithm finds a matched detection found nearby. When the tracker stays in the "lost" state for a certain amount of time without going back to the "tracked" state, it will be terminated by moving to the "inactive" state. The detail of the deterministic transitions of tracker's state in each camera is shown in the blue area of the flowchart, presenting our framework in Figure 4.2.

Concerning data association across cameras, after each camera independently performs its own tracking task, the trajectories of targets are gathered at a computation center in order to be associated with specific identities. As a result, the camera architecture optimally supporting this collaborative method is the star model as in Figure 2.27 a), except for the fact that instead of sending massive data of videos to the computation center, in our framework, each camera is sending only the data of trajectories of targets to the center. At the computation center, the trajectories from all views are associated with each other, and then the association result is sent back to each camera. Subsequently, the computational loads are separated on each individual camera (i.e., tracking tasks) and the computation node (i.e., association tasks). In terms of target-ID storage for the association at the computation center, we register the local ID of targets, which is the tracker number from each view, into an assignment table. An example of this assignment table is presented in the Tab. 4.1.

Identity	View 1	View 2	View 3	View 4
A	1	2	2	1
B	2	1	3	4
C	-	3	1	2
D	3	-	-	3

Table 4.1 – The data assignment in the computation node within our collaborative multi-camera framework. In this example, the identity A is being seen by all cameras with the tracker IDs $1, 2, 2, 1$ in views $1, 2, 3$ and 4, respectively. Meanwhile, the identity C is only being seen by views $2, 3$ and 4 (out of the FOV of view 1).

The assignment table yields the identities of targets to its tracker numbers on each view. As a consequence, the table also shows which identities are present, which are not in each view. The assignment of trackers across all cameras is sent to each camera for further processing, e.g., re-identifying lost targets or inferring targets' position during occlusion. Diagram 4.2 shows our multi-camera tracking framework. Indeed, the operations inside of the blue box are the loop of individually tracking tasks on single cameras at a single frame; meanwhile, the operations outside of the blue box are processed in the computation center.

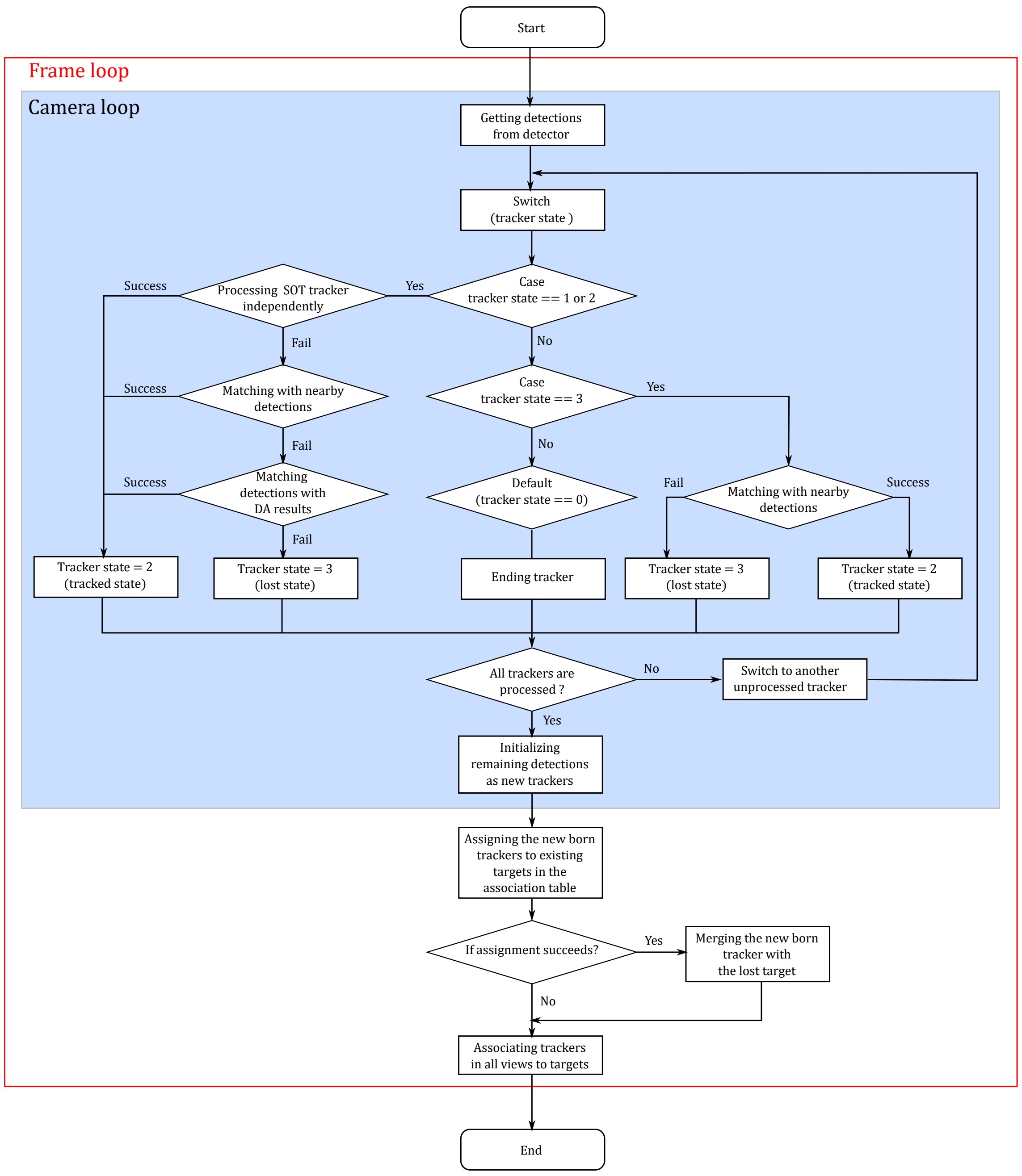

Start
Frame loop
Camera loop
Getting detections from detector
Switch (tracker state)
Case tracker state == 1 or 2
Yes
Processing SOT tracker independently
Success
Fail
Matching with nearby detections
Success
Fail
Matching detections with DA results
Success
Fail
No
Case tracker state == 3
Yes
No
Default (tracker state == 0)
Matching with nearby detections
Fail
Success
Tracker state = 2 (tracked state)
Tracker state = 3 (lost state)
Ending tracker
Tracker state = 3 (lost state)
Tracker state = 2 (tracked state)
All trackers are processed ?
No
Switch to another unprocessed tracker
Yes
Initializing remaining detections as new trackers
Assigning the new born trackers to existing targets in the association table
If assignment succeeds?
Yes
Merging the new born tracker with the lost target
No
Associating trackers in all views to targets
End

Figure 4.2 – Diagram of our multi-camera collaborative tracking approach

In the next section, we describe our data association method that matches the tracking results from individual cameras together to obtain the assignment table.

4.1.2 Deployment of the across-view association data to improve robustness

In this section, we explain the use of the multi-view assignment information to boost the performance of the tracking process in single views. Therefore, the association of targets from all views is useful in helping a single view trackers to handle occlusion issue. We localize the identity of the target on the ground plane, and then we exploit it by sending this information back to any camera which is failing to track due to occlusion. We separate this multi-view solution into two scenarios: the first is the partial occlusion case, and the second is the hard occlusion case. In the first case, the target is detectable but impossible to be associated with any tracker. By comparing the detection of the target with the targets' positions in other views, it is possible to re-link this detection to the correct target on the view when the occlusion is happening. This is useful to keep cameras tracking their targets in order to prevent them from losing their identities. In the second case of occlusion, the targets have completely disappeared from the scene, the target's record on multiple view allows to re-identify it once it reappears, and preserve its identity.

4.1.2.1 Partial occlusion cases

This section discusses partial occlusion cases that frequently occur in tracking videos with complex environments. The partial occlusion case is defined as when a target is detected, but cannot be identified among current trackers under tracking process, because *part* of the target is being hidden by other targets or by the environment. As a self-recovery tracking process, the association step tries to associate the target to a nearby detection in the case of SOT process failure. Indeed, matching characteristics between its templates and its detection usually fails, because only a part of the target is being matched, while the other part is being occluded. Hence the tracking process is interrupted, thus losing the target, i.e., identity lost.

To keep taking advantage of detection results, we propose an additional step to recover identity when a partly occluded target cannot be either tracked or associated with its detection. Concretely, from the views in which the target is being occluded, the representative points of all detections are projected into the common ground plane. These representative points are the middle points of the bounding box's bottom lines as the estimates of feet positions. Therefore, all the targets belonging to the missing identity from all views give their position on the ground plane to get an average position of the target. The detection, which is closest to this position and not further than a fixed threshold, will be added to the result of the tracker, which missed its target. These techniques allow MOT trackers to perform with efficiency in the case of the partial occlusion illustrated in Fig. 4.3.

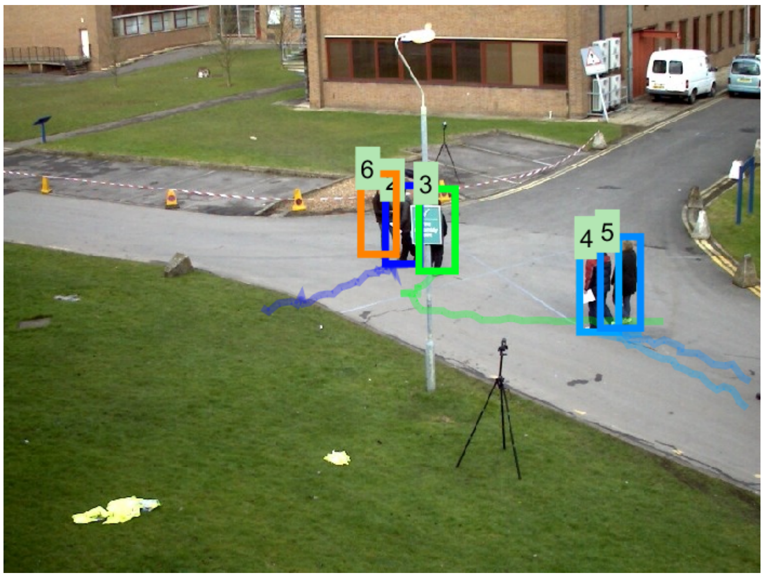


Figure 4.3 – The identity 3 is occluded in the displayed view because of the road sign. In this case, the person with identity 3 is detectable, but recognizable in the actual view. By using multiple cameras, our method is able to reassign the detection to the correct identity.

4.1.2.2 Long-time occlusion cases - Target re-Identification

Besides the partial occlusion, the long-time occlusion is frequently present in tracking videos. In this section, we explain our approach based on a multi-camera association to recover the identity of targets. The association strategy in a single view (as in Fig. 4.2) can handle short time occlusions when a target reappears at a position that does not exceed the search area around its last position within a given time. This is a common strategy used by many online trackers, as it can quickly recapture targets and resume tracking processes if targets' movements are simple. In this case, expanding the search area or/and the searching time period are possibly the solutions. They nonetheless do not suffice to reclaim all lost targets in any tracking scenario, in particular, when the time of disappearance varies and depends on each target and when targets are moving at different speeds with complex and unpredictable trajectories, then it is infeasible to infer where targets are going to reappear after occlusion.

To address this issue, we propose an approach using a network of cameras with *overlapping fields of view* in order to allow trackers from each individual view to recapture their lost targets by benefiting from the across-view association data. Precisely, when a target appears in the FOV of a camera, there can be three possibilities. Firstly, a target reappears after being completely occluded in a long time. Secondly, a target reenters to the scene where it has left before. Finally, a completely new target appears. In the last case, initializing a new tracker to pursue the new individual is appropriate. However, in the two first cases (i.e., occlusion consequences), re-identifying the target is reasonably expected rather than creating a new tracker. In our implementation, whenever a tracker is just born and successfully runs in several frames, which is confirmed not to be the false positive, it is considered as a newborn target. As illustrated previously (Fig. 4.2), for each "lost" target, its estimate from other views (via association results) is compared with all new targets. In detail, for each "lost" target, we collect the positions of the target from other views, called the position estimate of the "lost" target. Among all new born targets, we then seek a new born target, which is closest to the "lost" target's position estimate. This new born target is considered as belonging to the "lost" target, if and only if the average Euclidean distance between this new born target and the position estimate of the "lost" target is smaller than a pre-defined threshold. With this technique, our framework uses the position of "lost" targets across cameras within the network to re-identify it in the view which has been losing the target. The details of the ID-recovery algorithm are presented at Alg. 4. The entire algorithm of our multi-view multi-object tracking is shown in Alg. 3. In the next section, we detail the distance functions which are used to measure the affinity between two targets.

```
Input  : Set of video sequences from $K$ views. Object detection $\mathcal{D}_k = \{d_m^k\}_{m=1}^Q$
         for camera $k$.
Output: Trajectories of targets $\mathcal{T}^k = \{v_i^k\}_{i=1}^M$ in the $k^{th}$ camera, for all
         $1 \leq k \leq K$.
1  Initialization:  $\mathcal{T}^k \leftarrow \emptyset$, $1 \leq k \leq K$.
2  // main loop
3  foreach frame number $l$ in videos do
4      foreach each view $j$ do
5          // process targets in tracked states
6          foreach tracked target $v_i^k$ in $\mathcal{T}^k$ do
7              Follow the policy, move the MDP of $v_j^k$ to the next state;
8          end
9          // process targets in lost states
10         foreach tracked target $v_i^k$ in $\mathcal{T}^k$ do
11             Recover tracked state if found any similar detection covering the target;
12             Add the nearest detection into target if found in one of the subgraphs;
13         end
14         Data association for the lost targets;
15         foreach lost target $v_i$ in $\mathcal{T}^k$ do
16             Follow the assignment, move the MDP of $v_i^k$ to the next state;
17         end
18         Initialize the new targets from detection $d_m^k$ not covered by any tracked
             target in $\mathcal{T}^k$;
19     end
20     Update the set of subgraphs with the Algorithm 5 in Sec. 4.2 or Unbalanced
         Optimal Transport matching in Sec. 4.3;
21     Connect the newborn tracklets $v_{i'}^k$ to the lost target $v_i^k$ with the Algorithm 4;
22 end
```

Algorithm 3: MDP-based multiple-camera MOT algorithm.

```
Input  : Set of “lost” target from $K$ views $\mathcal{T}_{lost} = \{v_i^k\}_{i=1}^{M}$, Set of new targets
         $\mathcal{T}_{new} = \{u_j^k\}_{j=1}^{N}$. Set of subgraphs $\{G_s\}$
Output: A new set of “tracked” target $\mathcal{T}_{tracked} = \{v_i^k\}_{i=1}$, the “lost” target set
         $\mathcal{T}_{lost} = \{u_i^k\}_{i=1}^{m \leq M}$ and the new target set $\mathcal{T}_{new} = \{u_i^k\}_{i=1}^{n \leq N}$
1  // process targets in tracked states
2  foreach lost target $v_i^k$ in $\mathcal{T}_{lost}$ do
3      foreach new target $u_i^k$ in $\mathcal{T}_{new}$ do
4          Get the list of associated targets of $v_i^k$: $G_s$.
5          Measure the distances between $u_j^k$ and $\{v_i^{k' \neq k}\} \in G_s \setminus \{v_i^k\}$:
            $\{d\} \leftarrow \left\{ f_{dist}\left(u_j^k, v_i^{k'}\right) \right\}$
6          if Average distance $\bar{d} < \gamma_1$ then
7              Merge $v_i^k \leftarrow \left\{ v_i^k, u_j^k \right\}$
8              Discard $u_j^k$.
9          end
10     end
11 end
```
Algorithm 4: Algorithm of recovery identity of “lost” target.

4.2 Data association across cameras

This section describes our target association approach, which benefits from the Markov Processes of SOT trackers in the MOT framework [229]. Consequently, all *tracked* and *lost* trackers in each camera are assigned to those in other cameras. Each individual/target on the tracking scene is being tracked by a tracker on each camera. Based on its appearance and position, the trackers tracking the same target across cameras are associated together. On the other hand, the state of targets, e.g., *tracked* or *occluded*, is interpreted via associated trackers across cameras, this allows our multi-camera tracking framework to leverage these assignments to correct and recover the identity of targets after occlusion in each camera, hence to obtain more accurate tracking results in terms of ID-preservation.

4.2.1 Proposed methods

As an essential part of our multi-camera framework described in the last section, we present in this section, our approach of data association across cameras, which connects trackers found in all cameras to identify target individuals. Indeed, the input to our algorithm is a set of videos obtained by multiple *calibrated* and *frame-synchronized* cameras, whose FOVs significantly overlap. As mentioned in the previous section, the MDP of [229] for multiple object tracking in a single view is embedded in our multi-camera framework. Initially, after each camera obtained its own tracking independently, all the trackers from cameras are the input of our data association algorithm. Because any *target* found during tracking videos is presented by a *tracker*, for convenience, the term “target” indicates for “tracker” from now on. Basically, we only consider, *at each time t*, the set of “alive” targets, that is, the set of targets whose state is "tracked" or "lost", as inferred by the policy learned in [229]. Our goal is then to associate these targets across the different cameras. We formulate this goal as an optimization problem involving an undirected weighted graph. To keep the notation uncluttered, we drop, in the sequel, the dependency on the time t. It is however important to keep in mind that the data association algorithm we describe is applied sequentially through time.

We use the notation $G = (V, E, w)$, where V, E and w respectively denote the set of nodes, set of edges and the weights of the edges. In our formulation, the "alive" targets *of all cameras* are considered as the set V of nodes. Let $C_k = \{v_1^k, ..., v_M^k\}$ be the cluster of targets in the camera k. The cluster C_k includes, in particular, all detections which are not considered as being false positives. The edges of the graph are defined as $E = \{(v_m^k, v_n^l)|m, n \geq 1 \text{ and } k \neq l\}$, with the condition $k \neq l$ indicating that two nodes in same camera cannot be connected. A node v_m^k has a *trajectory feature-vector* $\mathbf{x}_m^k = \{x_m^{k,1} ..., x_m^{k,F}\}$, where $x_m^{k,i}$ corresponds to the 2-dimensional coordinates, on the 3-D world ground plane $z = 0$, in the previous i frame. The number F corresponds to the number of past frames retained. The coordinates on the plane xy (i.e., the common coordinate system) are obtained by projecting the tracking result from image plane by using the homography matrix obtained from calibration data.

A node v_m^k also has a bounding box Φ_m^k that will serve to compute *appearance feature-vector*, to be detailed later. The weight of an edge between two nodes is then defined by the sum of an appearance distance f_{app} and a trajectory distance f_{traj} as the following equation:

$$w(v_m^k, v_n^l) = \alpha\, f_{app}(\Phi_m^k, \Phi_n^l) + \beta f_{traj}(\mathbf{x}_m^k, \mathbf{x}_n^l), \tag{4.1}$$

where α and β are two coefficients to balance the contribution of two distance metrics, which is explained in the Sec. 4.2.3.

The computation of the distance function f_{traj} involves the trajectories between two targets in the last L frames:

$$f_{traj}(\mathbf{x}_m^k, \mathbf{x}_n^l) = g\left(x_m^{k,F-L+1:F}, x_n^{l,F-L+1:F}\right), \tag{4.2}$$

where $L = \min\left(\min\left(|\mathbf{x}^k|, |\mathbf{x}_n^l|\right), 20\right)$ and g is a function, to be detailed later, that quantifies the *average distortion* between trajectories.

As mentioned in [236], the process of matching a target in different views requires identifying correspondences of the target in all different views. Hence, the solution of the problem can be described as a connected subgraph of G in which each node (target) is selected from only one cluster (view). Therefore, the subgraph for a particular tracked person can be denoted by $G_s = (V_s, E_s, w_s)$. The set of nodes V_s has a general form $V_s = \{v_m^k | k \in \{1, ..., K\}\}$, $E_s = \{E(p, q)|p, q \in V_s\}$ and $w_s = \{w(p, q)|p, q \in V_s\}$. Fig. 4.4 shows some examples of connected subgraphs representing some targets through the views. By definition, since there is no edge within the same cluster C, we obtain a subgraph in the figure (its edges include all dashed and solid lines). Additionally, we impose a maximal distance constraint on the edges to ensure that targets on all views must get close enough to confirm the identity of an *unique* person. Precisely, given a particular target in a particular view, we eliminate all targets whose distance to the chosen one is greater than a ϵ value.

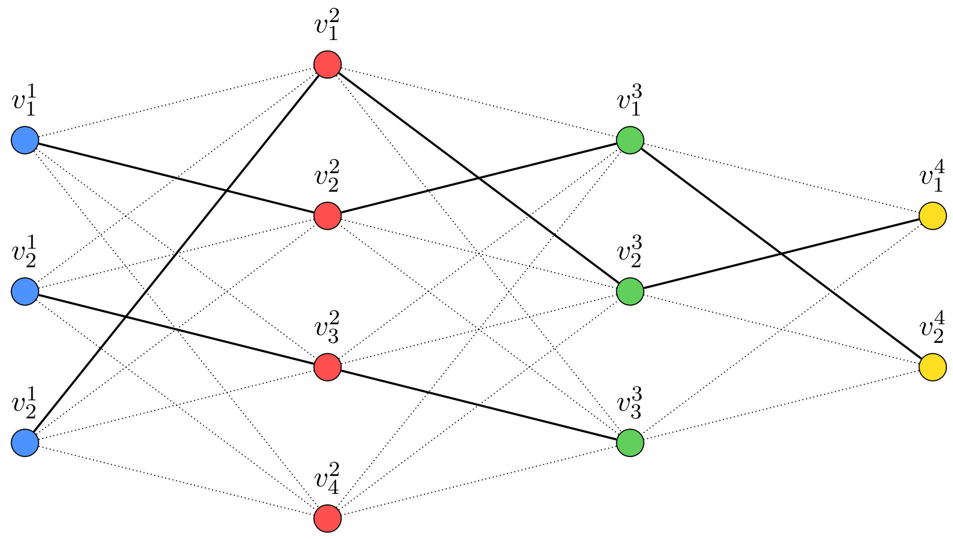


Figure 4.4 – Finding the corresponding targets in different views. For visualization purposes, we draw only edges between 2 successive views. We however stress that edges connect targets between *all* the views. Edges in solid line connect a target corresponding to the *same identity*, in different views. In this example, there are three connected subgraphs indicating three people in the tracking process [132]. An example of the subgraph of a target can be seen as the Fig. 4.5.

Associating targets across cameras now amounts to finding a connected subgraph having a maximum number of targets gathered from all views, and having the following minimum cost:

$$\min_{G_s \subset G} \mathcal{F}(G_s) = \sum_{m,n,k,l} w(v_m^k, v_n^l) \; s.t. \; k \neq l, \tag{4.3}$$

where

$$G_s = \{V_s, E_s, w_s | \forall (v_p, v_q) \in E_s, w(v_p, v_q) \leq \epsilon \in \mathbb{R}_+\}. \tag{4.4}$$

Since the problem is related to finding cliques in a given graph, the authors of [236] use the Generalized Minimum Clique Graph (GMCP) algorithm for matching detections through time. In our study, we do not consider the data association *through time*, but across cameras, at *each time*. We thus find a subgraph corresponding to each identity *in every frame*. In particular, we do not need to enforce temporal constraints, as was proposed in [236].

Since finding an optimal solution for our problem is related to finding a solution for the Travelling Salesman Problem (TSP) [236], we devise a simple, fast heuristic technique targeting an approximate solution well suited for the tracking application we consider. Instead of comparing all pairs of nodes, we select a node v_0 in G and include in the subgraph G_s all other nodes that are adjacent to v_0 and whose weight is less than a fixed threshold. In Alg. 5, we fix a camera k_0 in which all targets are selected as the reference nodes v_0 to perform the data association step. The comparison between our approximate solution and the optimal one is illustrated in Fig. 4.5. The details of the proposed algorithm are described in Algorithm 5.

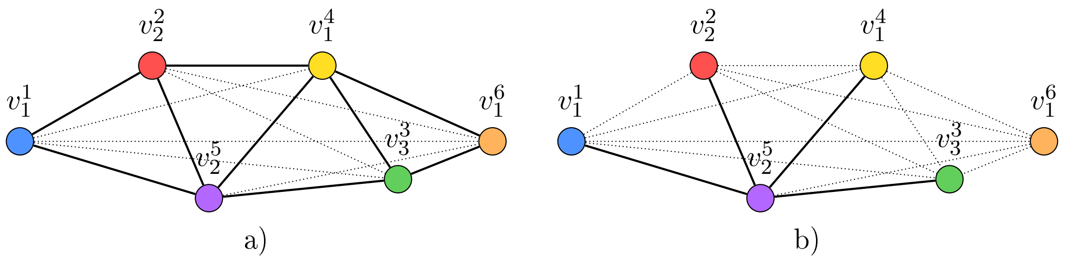


Figure 4.5 – Comparison between the optimal solution (a) and our fast heuristic (b). Note that node v_1^6 is included in the optimal solution, but not in the approximate one, because v_1^6 is not adjacent to the fixed node v_2^5 [132].

```
Input  : $K$ overlapping and frame-synchronized cameras.
         Set of tracked or lost targets $\mathcal{T}^k = \{v_i^k\}_{i=1}^M$ in all views
         A chosen *fixed* view $k_0$.
Output: $\mathcal{C}$ = set of subgraphs $G_s, 1 \leq s \leq N$, $N \leq M$, each indicating a given
         identity across views.
1  Initialize all subgraphs of $\mathcal{C}$ to the empty set.
2  foreach $v_i^{k_0}$ in view $k_0$ do
3      foreach each view $j \neq k_0$ do
4          Register ← $\emptyset$;
5          foreach $v_{i'}^j$ in view $j$ and $v_{i'}^j \notin G_j$ do
6              Compute the weight of the edge connecting targets $v_i^{k_0}$ and $v_{i'}^j$ (by eq.
                4.1);
7              Save the edge $(v_i^{k_0}, v_{i'}^j)$ and its weight to the Register variable;
8          end
9          Optimal cost ← $min(register)$;
10         switch setting do
11             case setting1 do
12                 condition = Optimal cost < $\epsilon_1$;
13             end
14             case setting2 do
15                 condition = Optimal cost < $\epsilon_1$ & $f_{traj}(v_i^{k_0}, v_{i'}^j) < 3.5\mu_1$
16             end
17         end
18         if condition then
19             Add $v_{i'}^j$ to subgraph $G_i$.
20         end
21     end
22 end
```

Algorithm 5: Algorithm for data association across cameras.

The next section describes the distance functions used in our framework to measure the affinity between targets in different cameras in order to associate targets across cameras, which address the occlusion problem in single-view and promote the ability to re-identify targets in the camera networks.

4.2.2 Distance functions

In this section, we discuss the features and the distance functions used in our algorithm in order to associate targets across cameras. As we mentioned previously, the appearance of targets is an important cue to determine whether two targets in two different views are identical or not. Generally, the algorithm accesses the appearance cues of a target through an image cropped from its bounding box. This image patch usually contains both the target and the background, and its appearance can differ from one camera to another, e.g., different color patterns, background, or changing shape from different points of view. In order to increase robustness, the affinity measure also includes a measure of the trajectory of targets which is the distance between targets' trajectories. This affinity measure is consistent and invariant in different views, since each target moves on a unique path in a specific period of time during the videos. We now present a variety of features and distances used in our algorithm and dedicate a subsection to the delicate issue of combining distances based on different features.

4.2.2.1 Trajectory-based features

This section discusses the trajectory-based features used to compute the affinity between any pair of targets of different cameras. We introduce a *distance metric*, which is generally a temporal signal comparison to measure the disparity or similarity between two-time series, which are the paths of targets *in different views* in our case. As mentioned previously, in a calibrated, synchronized, and overlapping camera network, to share the tracking results across cameras, all the trajectories of targets from all views must be converted and projected on a *common* plane. Therefore, the trajectories of targets we considered are the series of coordinates of targets, which are recorded during their movement, on the common *ground-plane*. In the literature, there are two common *distance metrics* to compare two temporal signals, which are different from each other by their data point matching methods: the pointwise Euclidean matching and Dynamic Time Warping (DTW) matching methods [185]. Firstly, the most basic and simplest method is the pointwise Euclidean matching, which is the matching each single data point of a signal to that of the other signal provided that those data points recorded at the same time. The second technique of matching, DTW, was introduced by Sakoe et al. [185] to cope with the difference of 2 similar signals recorded at different speeds. DTW is an effective matching tool that is used in many different applications, such as comparing stock trading data over similar time frames in financial markets or comparing audio clips for speech recognition. In general, it is a method that calculates an optimal non-linear match between two given data sequences with the objective to obtain the minimal cost, which is computed as the sum of absolute differences between matching points. The difference between these two matching methods is illustrated in Fig. 4.6.

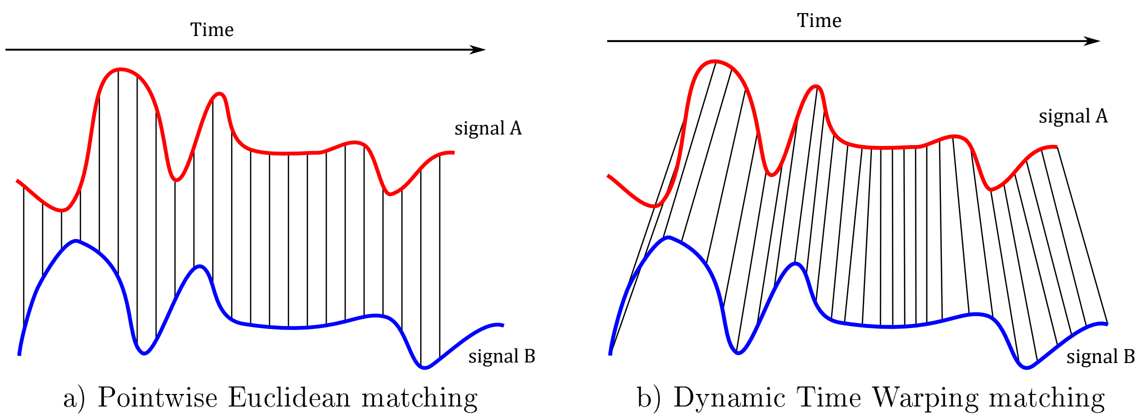

Time
signal A
signal B
a) Pointwise Euclidean matching
Time
signal A
signal B
b) Dynamic Time Warping matching

Figure 4.6 – Comparison between the pointwise matching and Dynamic Time Warping matching. Source: Wikipedia

- **Pointwise distance.** We consider trajectories involving L points, the point-wise distance between the trajectories of two targets is defined as the average value of the Euclidean distance between two positions of the targets at frame t. The formulation of the point-wise distance is written as follows:

$$f_{traj}(\mathbf{x}_m^k, \mathbf{x}_n^l) = \frac{1}{L}\sum_{i=0}^{L-1}\left\| x_m^{k,F-i} - x_n^{l,F-i}\right\|, \tag{4.5}$$

  where $\mathbf{x}_m^k = \{x_m^{k,1} ..., x_m^{k,F}\}$.

- **Dynamic Time Warping distance.** One of the issues in multiple-camera tracking systems is the *imperfection of frame synchronization*. It implies that the movement of targets might be recorded at slightly different time frames through different cameras. In reality, synchronizing frames across cameras means matching the frames recorded in a camera to those of other cameras in the system, such that the differences of recorded time between any pair of frames do not exceed a certain value, e.g., 10 ms, for example. There is always some tiny frame asynchronization in synchronized multi-camera systems. As a result, the path of a target might be dissimilar in different camera views. The Dynamic Time Warping algorithm allows the tracking process to alleviate this issue. We define DTW distance between 2 trajectories with length L as the following formula:

$$f_{traj}(\mathbf{x}_m^k, \mathbf{x}_n^l) = \frac{1}{L} d_{DTW}(\mathbf{x}_m^{k,F-L+1:F}, \mathbf{x}_n^{l,F-L+1:F}). \tag{4.6}$$

  Notice that DTW function d_{DTW} gives the sum of absolute differences of matching points, so to make this metric comparable to pointwise distance, we divide this value by the considered length L of the two input signals.

In the next section, we explain another type of features technically based on appearance of targets beside the trajectory-based features discussed above.

4.2.2.2 Appearance features

According to the literature on appearance features (in Sec. 2.5.4), which is commonly used in re-identification applications, there are two main classes of appearance features: handcrafted and learning-based features. In this section, we detail the appearance features that we used in our multi-camera tracking algorithm. In terms of handcrafted features, the histogram of colors is first the most popular feature in computer vision to characterize visual objects. Secondly, the backward-forward metric based on Lucas-Kanade matching points is used in LK tracker [27]. The final handcrafted feature we use is deep matching pairwise feature [221]. Alternatively, to extract the most relevant characteristics of targets, we apply two CNN-based learning appearance features as well.

- **Histogram of colors:** Color is an informative cue that allows us to distinguish between different targets. One of the most common features based on color is *Color Histogram* (CH), which is basically a multi-histogram of color pixel intensity of images, which can be a single channel (i.e., gray image) or multi-channel (e.g., RGB, HSV, CMYK). Besides the simplicity of computation of CH, several adverse effects of using CH, such as the target color change from camera to camera due to different image sensors of cameras, or the remaining portion of background in the cropped image leads to the confusion of targets and deteriorates tracking performance. In our implementation, after evaluating the histogram of all three RGB channels of the image extracted from the target's bounding box, we simply concatenate them into one feature vector. The distance metric is defined as the Euclidean distance between the CH vectors of two targets. The formulation is written as follows:
$$f_{app}(\Phi_m^k, \Phi_n^l) = f_{Euclidean}(\Phi_m^k, \Phi_n^l) = \left\| f_{CH}(\Phi_m^k) - f_{CH}(\Phi_n^l) \right\|_2 . \tag{4.7}$$

- **Lucas-Kanade Backward Forward error:** We adopt the appearance feature for object tracking presented in the paper of Xiang et al. [229], which is the combination of the optical flow matching and the Forward-Backward error. In general, the algorithm performs the matching between the densely sampled points of optical flow calculated from the target's current appearance and the considered detection in the next frame. From the corresponding point, the algorithm then performs another optical flow matching backward from points on the next frame to the those on the current frame. Concretely, on the current frame image I_t, an optical flow is computed from densely and uniformly sampled points inside the template to the next frame image I_{t+1}. Given a set of points $\{\mathbf{u}_i\} = \{(u_x, u_y)_i\}$ sampled inside the target template. Then the iterative Lucas-Kanade method with pyramids [27] operates on those points to seek their corresponding $\{\mathbf{v}_i\} = \{\mathbf{u}_i + \mathbf{d}_i\} = \{(u_x + d_x, u_y + d_y)_i\}$ on the next frame image I_{t+1}, where the set of vectors $\{\mathbf{d}_i\}$ are the optical flow of the target template (Fig. 4.7 (b)). As a result, the optical flow matching allows the tracker to predict the target's position in the next frame $t+1$. Notwithstanding, the matching is not always consistent with tracking, as illustrated in Fig. 4.7 (c). In order to decide whether accept or reject the prediction, the Backward-Forward (BF) error by Kalal et al. [116] measures the stability of the prediction. Indeed, let a forward optical flow be a set of points $\{\mathbf{u}_i\}$ on the current frame t and its correspondences $\{\mathbf{v}_i\}$ on the next frame $t+1$, and a backward optical flow be the points

$\{\mathbf{v}_i\}$ on the next frame $t+1$ and their correspondences $\{\mathbf{u}'_i\}$ on the current frame t. In case of reliable predictions, on the current frame image I_t, any pair of $\mathbf{u}_i$ and $\mathbf{u}'_i$ should be close to each other. Hence, the FB error at any point $\mathbf{u}_i$ is measured by the Euclidean distance between its location and that of its Backward-Forward prediction: $e(\mathbf{u}_i) = \|\mathbf{u}_i - \mathbf{u}'_i\|^2$. Finally, as an appearance distance [116], the stability of matching is measured by the median of the FB errors of all sampled points: $e_{medFB} = median(\{\mathbf{u}_i\})$.

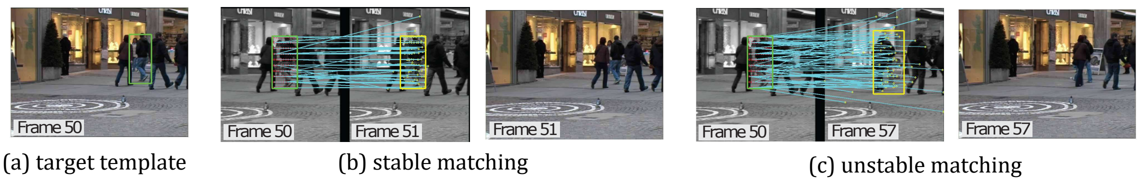


Figure 4.7 – The appearance of the target is represented by a template in a video frame (a). In figure (b), a set of sampled points $\{\mathbf{u}_i\}$ inside the target in frame 50 matches to a set of points $\{\mathbf{v}_i\}$ in the frame 51. The optical flow is computed from densely sampled points inside the target template to a new frame. The quality of the flow is used as a cue to make the decision: (b) an example of stable prediction, meanwhile the divergence of matching point caused by the unstable prediction (c). The yellow box is the predicted location of the target. [229]

Similarly, we define an appearance distance based on the median Backward-Forward error between two target appearances Φ^k_m and Φ^l_n in different views:

$$f_{app}(\Phi^k_m, \Phi^l_n) = e_{medBF}(\Phi^k_m, \Phi^l_n), \tag{4.8}$$

where e_{medBF} is the median of Backward-Forward errors, as predefined above.

- **Deep matching:** Similarly to LK matching approach mentioned above, Weinzaepfel et al. [221] developed an efficient way to densely match points from one image to another, which named *deep flow* or *Deep Matching*. The *Deep Matching* algorithm computes the dense optical flow on the entire image. This method densely finds the matching points from an image to another. Thus the approach is devoted to the quantity of the matching points rather than their quality. From the perspective of tracking problems, it becomes more appealing when it is used to match detections that belong to the same person, the matching points on the body is more easily found than those on the background, as the scene behind changes permanently while the target moves. Some examples of Deep Matching results are shown in Fig. 4.8. Inspired by Deep Matching method [221], Tang et al. [201] introduced the Deep Matching pairwise feature, which basically measures the appearance similarity between 2 targets based on the number of matching points between two detections.

  In our implementation, when the Deep Matching algorithm [177] is deployed on two distinct views, we observe that the number of matching points decreasing dramatically, so the pairwise feature used in [201] becomes inaccurate. In order to tackle this issue, we propose a variant of Deep Matching inspired by the Lucas-Kanade backward-forward distance mentioned above. Instead of considering the median

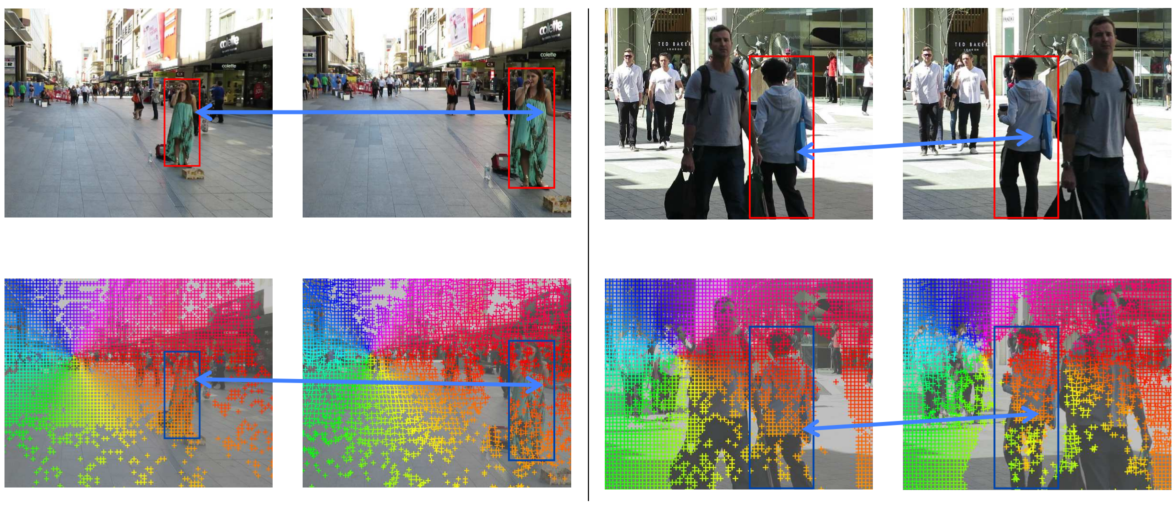


Figure 4.8 – Visualization of the Deep Matching results on the MOT16 sequences [201]. The keypoints (i.e., cross masker) having the same color are the pairs of deep matching between two images.

Backward-Forward error as the distance metric, we measure the average displacement of the matching points. Given the optical flow between two detections with K matching points, the matching points of the target's appearance $u_i \in R^2, 1 \leq i \leq K$ are matched to those $u'_i \in R^2, 1 \leq i \leq K$ on other detections in *other views*. Ideally, the matching between two distinct individuals will amplify the displacement on average, while the correct matching of the same identity on multiple views will result in a small displacement. Therefore, our Deep Matching distance function is defined as follows:

$$f_{app}(\Phi^k_m, \Phi^l_n) = e_{DM}(\Phi^k_m, \Phi^l_n) = \frac{1}{N}\sum_{i=1}^{N} ||u_i - u'_i||_2, \tag{4.9}$$

- **CNN-based learning appearance features:** As discussed in Section 2.5.4, the learning-based features show an impressive performance on representing images. Hence, we apply deep learning techniques to extract appearance features for the affinity measure between targets in our multi-camera framework. The current trend, fueled by the recent success in deep learning, considers the features corresponding to the output produced by the hidden layers of a deep neural network. As one of the first attempts to apply deep learning to extract appearance features for the re-identification task, Hermans et al. [105] introduced a novel CNN to extract useful appearance features, while eliminating extraneous factors such as background and moving body parts. In detail, the authors use the pretrained ResNet-50, replace the last FC layers by their own FC layers for their Re-ID task. The output of the CNN with the replaced FC layers embeds the input bounding box into an embedding space with the dimension of 128, where *Euclidean distance allows disambiguating different identities.*

  In the embedding space of appearance feature vectors, we can straightforwardly com-

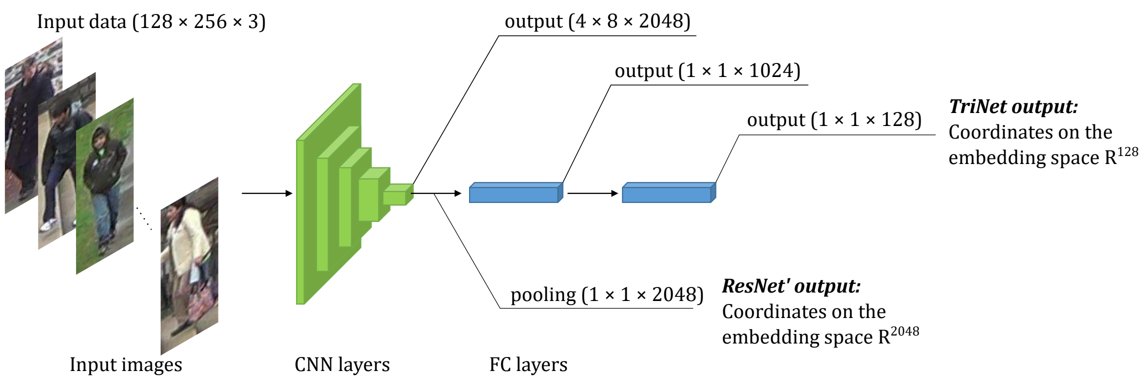


Figure 4.9 – CNN-based appearance feature extraction with our modified ResNet50 (ResNet') and TriNet.

pute the Euclidean distance between targets. Letting $F : \Phi_m^k \in S \to f_{net}(\Phi_m^k) \in R^n$ denote the learned embedding provided by the CNN, we can define an appearance distance as follows:

$$f_{app}(\Phi_m^k, \Phi_n^l) = \left\| f_{net}(\Phi_m^k) - f_{net}(\Phi_n^l) \right\|_2 . \tag{4.10}$$

In this study, we test the pretrained CNN [105], named triNet, and another CNN feature extractor, the resNet50 [99] without the last FC (fully connected) layers, named *ResNet'*. The architecture of CNNs we use to extract appearance features from target images is shown in Fig. 4.9.

4.2.3 Combining appearance and trajectory distances

In this section, we explain how the final distance metric is combined from the two types of distance metrics: appearance feature distance and trajectory-based distance. With the purpose of proposing a robust distance to be able to distinguish different individuals, we introduce an overall distance metric, which is a combination of appearance and trajectory distances, as written in the Eq. (4.1).

A crucial problem emerges when we observe the tracking results of any target across views of an overlapping and synchronized camera network. At any frame, a target always has different positions on the ground because those positions are from the projection of its on-view positions from different cameras on the ground plane. This disparity of positions between different cameras is caused by the calibration error or the asynchronization of frames of multiple cameras. Similarly, we assume that the cause of the disparity of appearance features of the same target across cameras is a type of camera network errors, e.g. color calibration error. we then characterize these types of disparities as the characteristic parameters of the camera network. To support this assumption, we conduct exhaustive experiments on two datasets including *PETS09-S2L1* and *terrace1*. Fig. 4.10 a) and c) show, for two different datasets, the (empirical) standard deviation of the ℓ_2 errors between the positions of the computed projections, seen from different cameras, and the ground truth. The figures show the evolution of these uncertainties through time (frames) and for each

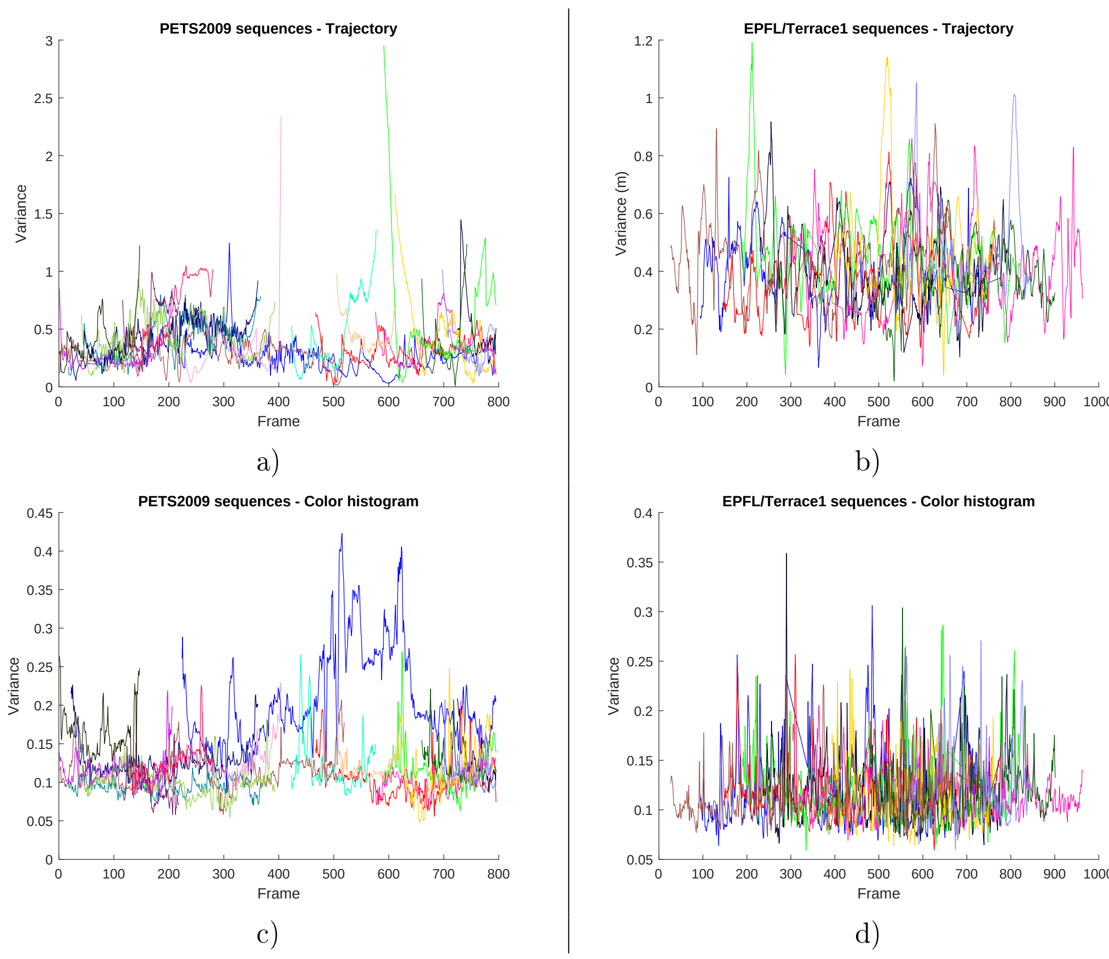


Figure 4.10 – The trajectory uncertainties in *PETS09-S2L1* sequence (a) and *terrace1* sequence (b). Figures (c) and (d) depict the uncertainties related to color histogram features. Each color corresponds to a particular identity.

particular identity. Fig. 4.10 b) and d) illustrate the same uncertainties, but this time, based on the ℓ_2 errors between the appearance features. The particular feature vector used in Fig. 4.10 b) and d) is the color histogram. These experiences show that different distance types (e.g. trajectory v.s. color histogram) require different thresholding values. Based on this finding, we propose to relate the values of α and β to these location and appearance uncertainties, respectively. More precisely, we consider the following distance:

$$w(v_m^k, v_n^l) = \frac{1}{2}\left(\frac{1}{\mu_1} f_{app}(\Phi_m^k, \Phi_n^l) + \frac{1}{\mu_2} f_{traj}(\mathbf{x}_m^k, \mathbf{x}_n^l)\right), \tag{4.11}$$

where μ_1 and μ_2 are the average, through all the identities present, of location and appearance uncertainties, as afore-defined.

The next section details the implementation of our multi-camera tracking approach and the experimental results to prove the robustness of our multi-camera framework.

4.2.4 Performance evaluation

In this section, we present a common benchmark to evaluate the performance of our Multi-camera Multi-Object tracking algorithm and our analysis of the experiments we conducted. Firstly, we use the two most important benchmarks consisting of MOT-CLEAR and ID measure, which are widely recognized by the Multi-Object Tracking community. Secondly, in the implementation, we specify two different settings of our algorithm, detection task, as well as the existing methods for comparison. Finally, to validate our multi-camera approach, we analyze the experimental results of our approach compared with those of other approaches. These analyses are based on many aspects, including the impacts of trajectory-based and appearance features on performance and the quantitative comparison between our method with different settings and other methods.

4.2.4.1 Implementation

We keep the same parameters of the original proposition of MDP [229], the others parameters in our approach are detailed below. For target association algorithm, we chose the view 1 as the dominant view k_0.

Setting. For experiments, we test our algorithm with two grouping conditions:

- Setting 1: $w(v_m^k, v_n^l) < \epsilon_1$
- Setting 2: $w(v_m^k, v_n^l) < \epsilon_1$ and $f_{traj} < \epsilon_2$,

where $\epsilon_1 = 4.2$ and $\epsilon_2 = 3.5\mu_1$, where μ_1 is the position uncertainty level, as defined in the section 4.2.3. In other words, we test our algorithm without and with a spatial distance constraint to analyze the contribution of appearance features in the tracking results.

Detection. In all Tracking-by-Detection approaches, the detector plays an important role in tracking performance. We employ the widely used, public ACF (Aggregate Channel Features) detector [69] on all views of the sequences “PETS09-S2L1” and “Terrace1”, using the pre-trained Caltech model [223].

Competing methods. To evaluate our approach, we compare our method with the original MDP single-camera method and the multi-camera K-Shortest path (KSP) from the state-of-the-art. The KSP algorithm outputs quantized positions in the probabilistic occupancy map (POM) [81]. Our implementation uses a grid of size 36×36. We also test our algorithm with all different trajectory distances and appearance features detailed previously:

- Pointwise and Dynamic Time Warping (DTW)
- Color histogram, median Backward-Forward LK error (LK), DeepMatching (DM) and learned appearance features using triNet and resNet50.

4.2.4.2 Tracking benchmark

Comparing the performance of different MOT trackers is a delicate issue. The choice of the metric generally depends on the user’s end objective. For security applications, the identity of the target is one of the most important criteria, while for monitoring crowds and traffic, the number of mismatches between the ground-truth and the predicted result is more appropriate. In our study, we evaluate our method by using the CLEAR MOT [22] and ID measure [178]. It is important to note that there is a huge difference between these two metrics. The CLEAR MOT metric evaluates a tracker based on how often mismatches, in covering target region image, happen, while the ID metric measures how long the tracker keeps the identity of targets. The latter measure relies on an offset boundary value Δ from the ground-truth. It counts how many times the tracker gives the prediction within the boundary.

In this section, we present our experimental results verifying the efficiency of our multiple-camera MOT algorithm. To evaluate MOT performance, the benchmark MotChallenge [133] has been released with 2 datasets (MOT15 and MOT16), which contain a number of single-camera video sequences recorded by static or dynamic cameras, and the evaluation metrics of CLEAR MOT [22] and ID measure [178] are used. Additionally, the MotChallenge also provides multi-camera video sequences, but with cameras with mostly non-overlapping fields of view. Unfortunately, these datasets do not fit to the setting we consider in this study. We thus test on different datasets, to be explained later, but we still evaluate the performance of the method by relying on the CLEAR MOT and ID family of metrics, since they are still appropriate for our current setting. Since our method was especially designed to improve identity robustness, we will emphasize ID scores in the sequel.

Datasets. We used the well-known PETS2009 [78] and EPFL Multi-camera Pedestrian Videos [20] datasets. Among all sequences of PETS2009, the most relevant and suitable for our multiple-camera tracking system is "PETS09-S2L1" with 7 views from 7 synchronized and calibrated cameras. For our experiment, we only used 1 main view (from the camera 1) and 4 close-up views (from the cameras 5, 6, 7, and 8). Meanwhile, the EPFL dataset provides multiple indoor and outdoor video sequences, recording pedestrians by 4 different cameras. Because of the similarity between sequence scenarios, we selected the sequence "Terrace1" for our experiments. Fig. 4.11 shows that about $15-20\%$ of the observable zones are covered by all cameras. The PETS2009 dataset initially does not provide the ground truth. Fortunately, MotChallenge[1] provided the ground truth and detections on view 1. To complete the multi-view dataset ground truth, we manually annotate all identities on the other views. The ground truth of the sequence "Terrace1" is already published. The detections of the other views of PETS2009 and those of "Terrace1" are obtained by the same public detector used on MotChallenge, which is detailed in the Section 4.2.4.1. The ground truth and detection data on all views will be published on our project page[2].

Evaluation metric. To validate the efficiency of our multiple-camera multiple-object tracking method, we adopt the CLEAR MOT metric and ID measures and in particular the following scores: MOTA (multiple-object tracking accuracy), MOTP (multiple-object tracking precision), IDs (identity switches), IDF1 (ID F1-score), IDP (ID precision), IDR (ID Recall), False Positive (FP) and False Negative (FN). For further details on the metric, we recommend the MOTChallenge website[3].

[1]https://motchallenge.net
[2]https://github.com/quoccuongLE/MDP_MTMC_Tracking
[3]https://motchallenge.net

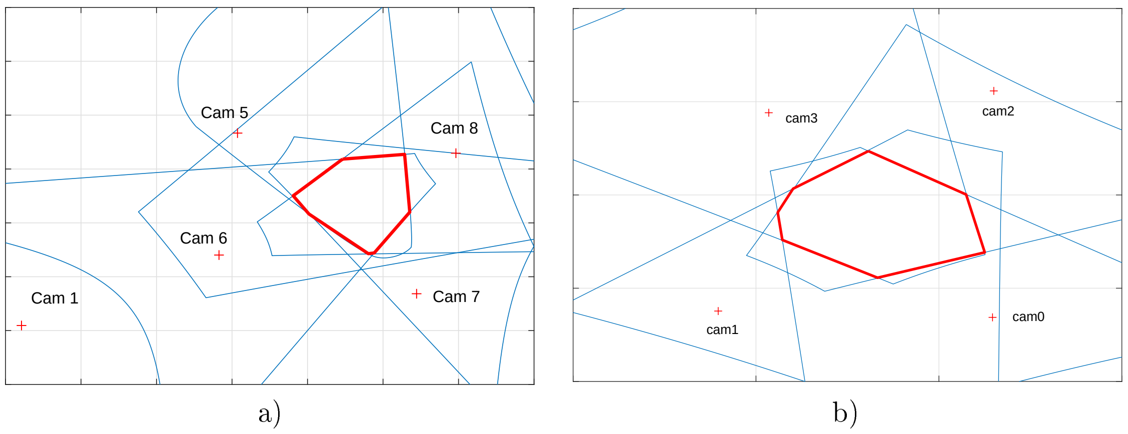


Figure 4.11 – Fields of view of the cameras in the PETS09-S2L1 sequence (a) and terrace1 sequence (b). The common overlapping zone has red contours. The rectangle unit has a dimension of 5 × 5 meters in the real world [132].

4.2.4.3 Experimental results

Overview. The results shown in the following tables and bar charts are the average values from all views. Concretely, the overall tracking results of the PETS sequence can be seen in the Table 4.2 and Table 4.3. Each score column has either an ↑ or a ↓ indicating whether better corresponds to higher or lower, respectively. In the bar charts, we only display the two most important scores, IDF1 and MOTA, with a horizontal blue line representing the reference IDF1 score of the original *single-camera* MDP method.

Primarily, our proposed method focuses on tracking targets in the hard occlusion case. It leads to an important reduction of identity switches and a significant improvement of ID measures. In detail, with only $15 - 20\%$ overlapping zone, on average in the best setting, our approach on the considered PETS sequence increased about 14.9% for IDF1, 14.3% for IDP and reduced 36.7% of ID switches. In terms of CLEAR MOT scores, our approach slightly improves both MOTA and MOTP scores. This can be explained by the fact that tracker's identification ability is not captured by the CLEAR MOT metric [178]. On the EPFL sequence, we observe the same effect. More precisely, our method remarkably improved the ID measure scores in the Table 4.4 and Table 4.5. The score increases by 27.7% for IDF1 and 27.2% for IDP. In contrast, the MOT scores and ID switch number slightly decreased, but these changes are not notable.

The KSP method performs poorly on the sequence PETS09-S2L1, but gives a high score on EPFL/terrace1. This behavior can be explained by the fact that the KSP method was developed on the EPFL multiple- camera Pedestrian Dataset. In fact, the authors assume that the targets have to finish their complete trajectories before leaving the scene. The in/out position of targets is also fixed on the scene, so we can see the actors walking in and out at the same place. On EPFL/terrace1, the algorithm found 8 paths that exactly correspond to the 8 targets in the video, thus leading to a high IDF1 score. Back to the sequence PETS09, the algorithm cannot deal with the targets that usually went out and then returned into the scene. It just found the longest paths and completely ignored the targets which appeared in a short period of time and regularly get confused by other targets at the boundary. Moreover, in the PETS09 database, there are no constraints on where people will appear and disappear on the scene. Consequently, KSP receives a negative score on MOTA. It indicates that the KSP algorithm cannot handle the enter/exit of targets. Another problem with KSP is that the tracking process occurs on a grid, called Probabilistic Occupancy Map (POM), whose unit size directly affects the accuracy of the tracker. Unfortunately, increasing the resolution of the POM required more iterations to make sure the occupancy map converged correctly.

Method	**IDF1**↑	**IDP** ↑	**IDs**↓	**MOTA**↑	**MOTP**↑
MDP	57.49	62.24	333	68.44	68.83
MDPmv + triNet	62.34	67.11	270	69.14	68.94
MDPmv + resNet50	62.35	67.15	250	69.54	68.84
MDPmv + LK	63.76	68.80	252	69.50	68.84
MDPmv + DM	58.97	63.75	260	68.88	68.94
MDPmv + CH	64.14	69.11	251	69.15	68.99
KSP	21.51	18.16	812	-29.63	64.27

Table 4.2 – Setting 1: Overall scores of MOT metric on “PETS09-S2L1” sequence.

Method	**IDF1**↑	**IDP** ↑	**IDs**↓	**MOTA**↑	**MOTP**↑
MDP	57.49	62.24	333	68.44	68.83
MDPmv + triNet	65.76	70.91	213	70.54	69.04
MDPmv + resNet50	65.93	71.09	212	69.94	68.91
MDPmv + LK	64.84	69.77	230	70.24	68.90
MDPmv + DM	66.00	71.00	207	70.67	68.99
MDPmv + CH	67.69	72.85	192	70.70	68.96
KSP	25.85	23.51	695	19.57	62.26

Table 4.3 – Setting 2: Overall scores of MOT metric on “PETS09-S2L1” sequence.

Method	**IDF1**↑	**IDP** ↑	**IDs**↓	**MOTA**↑	**MOTP**↑
MDP	12.29	16.36	656	47.14	72.65
MDPmv + triNet	15.46	20.60	736	46.57	72.42
MDPmv + resNet50	17.87	23.92	741	47.64	72.62
MDPmv + LK	15.76	20.87	768	46.49	72.53
MDPmv + DM	13.62	18.37	730	46.78	72.48
MDPmv + CH	16.00	21.12	761	46.85	72.44
KSP	25.85	23.51	695	19.57	62.26

Table 4.4 – Setting 1:Overall scores of MOT metric on “terrace1” sequence.

Analysis of different features. We test our approach with different appearance features. According to the IDF1 score, and excluding the KSP method, the results show

Method	**IDF1**↑	**IDP** ↑	**IDs**↓	**MOTA**↑	**MOTP**↑
MDP	12.29	16.36	656	47.14	72.65
MDPmv + triNet	14.41	19.08	710	46.57	72.42
MDPmv + resNet50	14.62	19.35	692	47.64	72.62
MDPmv + LK	14.93	19.85	681	46.49	72.53
MDPmv + DM	17.89	23.82	689	46.78	72.48
MDPmv + CH	16.60	21.98	699	46.85	72.44
KSP	25.85	23.51	695	19.57	62.26

Table 4.5 – Setting 2: Overall scores of MOT metric on "terrace1" sequence.

that there is no dominant appearance feature showing the best scores for all the tests. We remark that the LK feature and the two learned appearance features using triNet and resNet50 display a stable performance in the entire experiment. In contrast, the hand-crafted DeepMatching feature has inconsistent results with the two settings. Indeed, on the one hand, it performs poorly in Setting 1, but on the other hand, it reaches the performance level of the other features in Setting 2. Surprisingly, the color histogram appears to be a simple, stable, and good appearance feature, reaching high scores in all settings. The impact on the CLEAR MOT metric of the overall multi-camera algorithm, including all the features variants considered, is not significant. This was somehow expected as the main motivation behind the proposed method is to increase identity robustness. The rest of this section will thus concentrate on ID scores.

Comparison of different trajectory distances. We tested the two trajectory distance functions in Settings 1 and 2. The IDF1 mean value, concerning all appearance features, is shown in Table 4.6. The reported values indicate that Setting 2 notably helps the multiple-camera approach increase its performance, as opposed to Setting 1. We also realize that within the setting 2, our algorithm displays the same score with different appearance features. That means the additional spatial constraint, imposed in Setting 2, produces a stabilization with respect to other features.

Table 4.6 also summarizes the average scores, for all views, obtained when using the pointwise and DTW trajectory distances. The overall results indicate, in this case, that the improvement provided by DTW is not significant.

Sequence	**Path error**	**Setting**	**IDF1**
PETS09-S2L1	pointwise	1	62.31
		2	66.05
	DTW	1	62.77
		2	65.89
EPFL/terrace1	pointwise	1	15.74
		2	15.69
	DTW	1	15.69
		2	16.27

Table 4.6 – Comparison of Settings 1 and 2, and pointwise and DTW distances.

Qualitative comparison. Fig. A.6 provides a visual comparison of tracking results between our multi-camera approach and the MDP single-camera approach, on the sequence PETS09-S2L1. Overall, we can see that the multi-view system has the ability to recover the lost track actively by sharing the tracking results between cameras in the network. This explains the outperformance of our multi-camera tracking approach to the single-view approach. Visually, we observe that the trajectories obtained in the single-camera setting are notably shorter than those obtained by our multi-camera method. Also, the labels of tracked objects (see the labels in the last frame) are much higher in single-camera that in multi-camera mode. This is directly linked to the unique ability of the multiple-camera algorithm to recover to the state of targets. The detailed results of all experiments are shown in Appendix A.

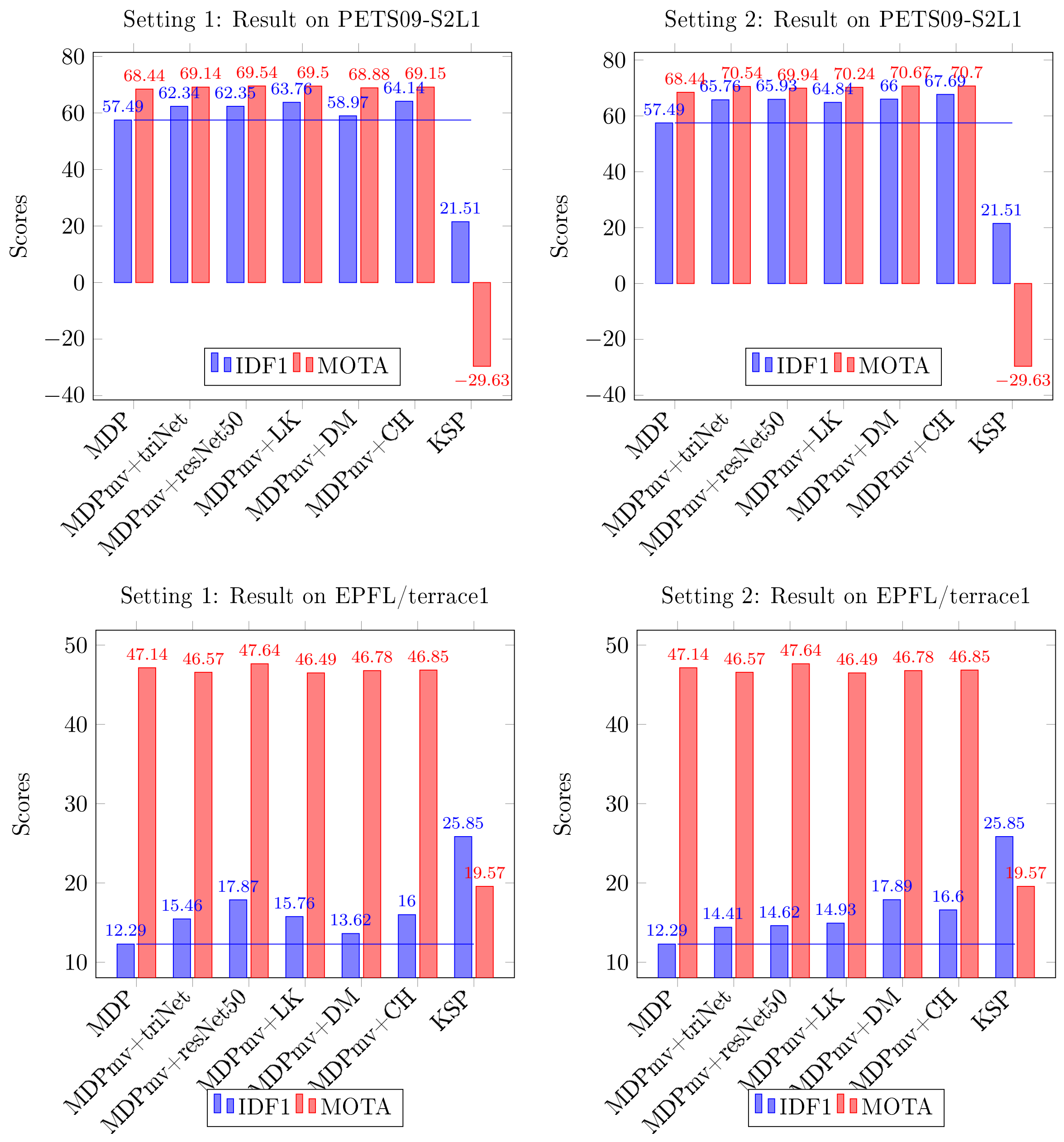


Figure 4.12 – Performance comparison between the reference single-camera method [229] and our multi-camera approach using IDF1 and MOTA scores.

4.3 Target Association via Unbalanced Optimal Transport

This section presents our second data association approach, which adopts the re-identification strategy in multi-object tracking within our multi-camera tracking framework. As we mentioned earlier in this dissertation, one of the interpretations of multi-object tracking is to identify it from detection sets gathered from camera videos. In our case study, we reformulate multiple object tracking problem across multi-view as assigning targets from one view to another. As a result, within our multi-camera tracking framework, target association occurs in pairs of cameras for further tracking collaboration. Indeed, targets being tracked on one view are not necessarily present in another view, then the target assignment between a pair of cameras is an assignment problem between two non-equal-size sets of targets. We treat this assignment problem as an unbalanced Optimal Transport on a feature space where the matching between projections of two sets can be more accurate. We develop a deep distance learning method based on optimal transport to project tracking target features, including appearances and locations on a feature space. The experiments of this approach are conducted on the same multi-camera tracking video datasets in the previous section. The experimental results show the effectiveness of our second association method in comparison with the method in the last section.

4.3.1 Motivation: Combining multiple distance features

Before describing our Optimal Transport assignment approach, in this section, we explain the necessary demand for utilizing both appearance features and trajectories of targets in the re-identification across different views during the tracking process, as well as the problems of combining those features. As the primitive idea of using all distances of features presented in the last section, we first investigate the performance of each feature individually in order to combine them later. To deeply analyze the impacts of each type of distance on tracking performance, the experiments in the last section are now replicated, but with only one type of distance for each experiment. Each identity has its own appearance, which is represented as a target on each camera; these targets are associated with their identity via the given ground truth data. At each frame, we already know correspondences of the targets across camera views, all feature distances between the targets across views of the same identities (i.e., *relevant elements*), as well as all the feature distances of those not belonging to the same identities (i.e., *irrelevant elements*) are straightforwardly computed. The whole process operates on all video frames to collect all samples. During the tracking process, as described in Section 4.2.1, given a primary target of a selected camera, we want to find all its corresponding targets of other cameras. On all other cameras, any target, whose distance with the selected camera's target is smaller than a value ϵ, is predicted as belonging to the same identity as the primary target's identity (i.e., *selected samples*), otherwise it does not (i.e., *unselected samples*). From these collections of samples, the precision and recall scores for each feature distance can be measured. We then construct the precision-recall curves of all feature distances, including their Area Under Curve (AUC) scores in Fig. 4.13.

In this experiment, we assume that our *only-path* multi-camera tracking process gives the "good-enough" tracking results. This is because most of the time, in the video sequences, targets are not occluded in the sense that the frames of "not-being occluded" outnumbering the frames of "being occluded". Therefore, tracklets obtained from the tracking algorithm are presumably not too short, then using their paths to identify targets is relatively accurate. Notice that this quantitative analysis aims to evaluate different appearance features with respect to the trajectory feature. The experimental results (Fig. 4.13) show that the trajectory or "path" feature is always ahead of all other appearance features with the AUC score of 0.951 in comparison with the second-best features (resNet50 [99]) $AUC = 0.219$. Both learned and hand-crafted appearance features, which describe the distinctiveness of target based on their appearance, are far less effective than trajectory. Overwhelmingly, this quantitative analysis implies that for multiple target tracking, the historical positions of targets, i.e., *trajectory* or *path*, is undeniably the most relevant and distinctive feature, which is actively showing the uniqueness of a target among others in different views of the camera network. However, the appearance of targets remains essential in many cases where the trajectories are relatively close or too short due to the tracking failure in single views. In those cases, the trajectory of targets appears as a weak feature for re-identifying them across cameras; meanwhile, those targets can only be distinguished by their physical appearance (e.g., clothes, body). Obviously, appearance features are essential in target association, but using raw appearances of targets from different cameras for data association is improbable and ambiguous, as mentioned in Sec. 2.5.4. Within a camera, between two frames, the appearance of a target seems unchanged, and the use of appearance features is straightforward. Between cameras, the variability of appearance for the same identity is much more evident.

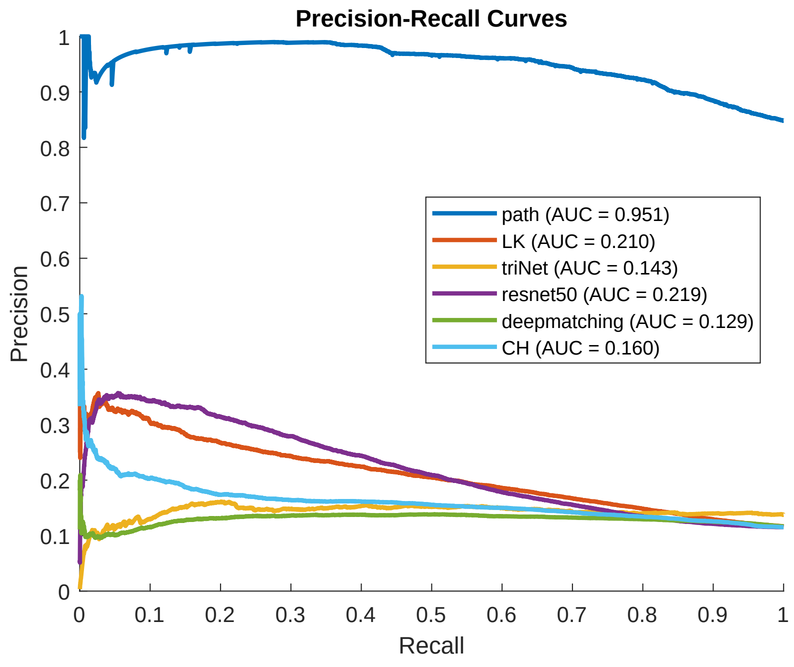


Figure 4.13 – The Precision-Recall curves of different feature for target clustering showing the outperformance of *trajectory* or *path* feature over all others which are based on appearance of targets.

This justifies our motivation to develop a novel target association method based on optimal transport, which takes into account the appearance variability across cameras in data association. The proposed approach genuinely puts together the positions and appearances of a target from different cameras on a unique feature space where the targets' embeddings are located closely together if they belong to the same identity, otherwise far away if their identities are different. We detail our proposed approach in the next section.

4.3.2 Proposed Targets Association Method

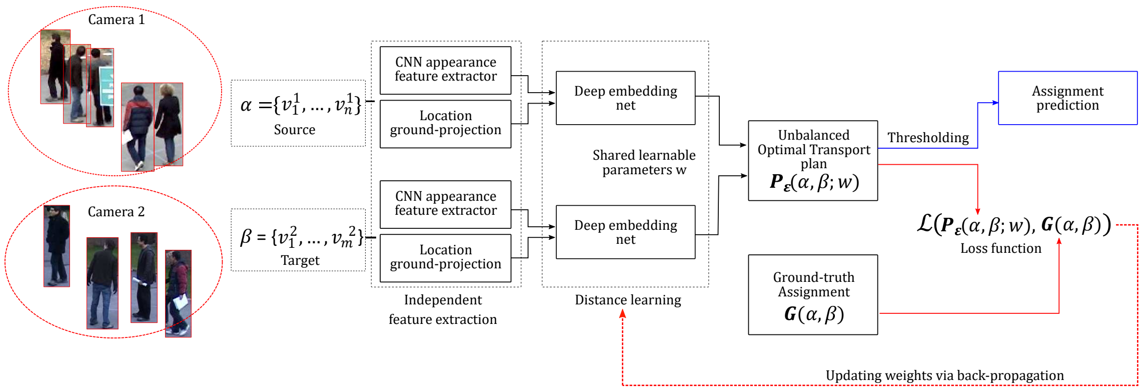

Camera 1
Camera 2
$\alpha = \{v_1^1, \dots, v_n^1\}$
Source
$\beta = \{v_1^2, \dots, v_m^2\}$
Target
CNN appearance feature extractor
Location ground-projection
CNN appearance feature extractor
Location ground-projection
Independent feature extraction
Deep embedding net
Shared learnable parameters w
Deep embedding net
Distance learning
Unbalanced Optimal Transport plan $\boldsymbol{P}_\varepsilon(\alpha, \beta; w)$
Thresholding
Assignment prediction
$\mathcal{L}(\boldsymbol{P}_\varepsilon(\alpha, \beta; w), \boldsymbol{G}(\alpha, \beta))$
Loss function
Ground-truth Assignment $\boldsymbol{G}(\alpha, \beta)$
Updating weights via back-propagation

Figure 4.14 – The pipeline of our distance learning framework. The red arrow indicates the direction during training process, meanwhile the blue lines for testing.

This section describes in detail our approach to solving target association across cameras problem via optimal transport. In the same context as Sec. 4.2, within a frame-synchronized overlapping camera network, associating targets between different cameras emerges as the main issue for collaborating multiple cameras to track multiple targets. Let's consider a pair of cameras C_1 and C_2 at a *single* frame. Given N targets $\{v_1^1, \dots, v_n^1\}$ in the camera C_1 and M targets $\{v_1^2, \dots, v_m^2\}$ in the camera C_2 with $n \neq m$, in general, the formulation of the problem is to match N targets from camera C_1 to M targets from camera C_2. Each target v_n^k is represented by an appearance feature Φ_n^k, which is extracted from frame image, and a position, which is the projection of the target's feet from frame image onto the 3D ground plane. We reformulate the target assignment within a pair of cameras as an Optimal Transport problem. The two sets of targets $v_i^{1^n}$ and $v_i^{2^m}$ are used to define two empirical distributions supported on a feature space $\mathcal{X}$.

$$\alpha := \sum_{i=1}^{n} \mathbf{a}_i \delta_{x_i}, \tag{4.12}$$

$$\beta := \sum_{j=1}^{m} \mathbf{b}_j \delta_{y_j}, \tag{4.13}$$

where δ_x is the Dirac at $x \in \mathcal{X}$ and $\mathbf{a}_i$ and $\mathbf{b}_j$ are the corresponding weights. In our setting, we will consider uniform discrete measures, which is, all the components of a weight vector are equal.

The *balanced* optimal transport problem involves measures α and β with the same total mass, usually taken, without loss of generality, to be probability measures. The Kantorovich's optimal transport problem is defined as follows:

$$\mathbf{L_C}(\mathbf{a}, \mathbf{b}) := \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle = \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a},\mathbf{b})} \sum_{i,j} \mathbf{C}_{ij} \mathbf{P}_{ij}, \tag{4.14}$$

where $\mathbf{C}$ is the ground cost matrix, whose elements $\mathbf{C}_{ij}$ are the pairwise distance between the Dirac δ_{x_i} of the source measure α and those δ_{y_j} of the target measure β, and $\mathbf{U}(\mathbf{a}, \mathbf{b})$ is a coupling from the *source* $\mathbf{a}$ to the *target* $\mathbf{b}$. The feasible couplings are defined by a set of coupling matrices $\{\mathbf{P} \in \mathbb{R}_+^{n \times m}\}$, where $\mathbf{P}_{ij}$ depicts the amount of mass flowing from x_i toward y_j, under the mass preservation constraint.

$$\mathbf{U}(\mathbf{a}, \mathbf{b}) = \{\mathbf{P} \in \mathbb{R}_+^{n \times m} : \mathbf{P}\mathbb{1}_m = \mathbf{a} \,\text{and}\, \mathbf{P}^T \mathbb{1}_n = \mathbf{b}\}. \tag{4.15}$$

When the discrete measures involved have large support, Problem (4.14) becomes computationally demanding. A now popular approximate solution has been introduced in [53]. It involves introducing a regularization parameter ϵ and considers the regularized problem which is defined as follows:

$$\mathbf{L}^{\varepsilon}_{\mathbf{C}}(\mathbf{a}, \mathbf{b}) = \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a},\mathbf{b})} \langle \mathbf{P}, \mathbf{C} \rangle - \varepsilon \mathbf{H}(\mathbf{P}), \tag{4.16}$$

where the discrete entropy of a coupling matrix is defined as follows:

$$\mathbf{H}(\mathbf{P}) = -\sum_{i,j} \mathbf{P}_{i,j} \left(\log(\mathbf{P}_{i,j})\right) \tag{4.17}$$

The Optimal Transport (OT) problem with entropic regularization (4.16) has a dual form following:

$$\min_{\mathbf{P}>0} \max_{(\mathbf{f},\mathbf{g}) \in \mathbb{R}^n \times \mathbb{R}^m} \langle \mathbf{C}, \mathbf{P} \rangle - \varepsilon \mathbf{H}(\mathbf{P}) + \langle \mathbf{a} - \mathbf{P}\mathbb{1}_m, \mathbf{f} \rangle + \langle \mathbf{b} - \mathbf{P}^T \mathbb{1}_n, \mathbf{g} \rangle \tag{4.18}$$

where the set of admissible dual variables (called *potentials*) ($\mathbf{f} \in \mathbb{R}^n, \mathbf{g} \in \mathbb{R}^m$). The optimal transport plan solved via the dual problem (4.18) has a closed-form [79]:

$$\mathbf{P}^{\star} = \pi = \exp\left(\frac{1}{\varepsilon}\left(\mathbf{f} \oplus \mathbf{g} - \mathbf{C}\right)\right) . \left(\alpha \otimes \beta\right), \tag{4.19}$$

where $\mathbf{f} \oplus \mathbf{g}$ is denoted as a sum matrix of 2 vectors f and g whose cell $\{\mathbf{f} \oplus \mathbf{g}\}_{ij}$ is equal to $\mathbf{f}_i + \mathbf{g}_j$, in the same manner, $\alpha \otimes \beta$ is also denoted as a product matrix of 2 vectors α and β whose cell $\{\alpha \otimes \beta\}_{ij}$ is equal to $\alpha_i \beta_j$. In our setting, our interest in the regularized version of OT, as described by Problem (4.16), does not stem from the fact that the involved discrete measures have a large support, i.e., the number of detections on each frame is on the order of 10 to 100. From our point of view, the interesting aspect is that the regularized OT implemented with the Sinkhorn algorithm makes it possible to adjust the ground cost for a particular application. More precisely, via automatic differentiation through the Sinkhorn iterations, the ground cost can be *learned* by minimizing an application-specific cost function. In literature, differentiating through the result of Sinkhorn iterations has recently been proposed in [85] for metric learning for generative modeling, and in [79] for approximating the gradient of the Sinkhorn divergence.

As an extension of Optimal Transport, unbalanced optimal transport does not require the mass preservation from *source* to *target*. In the formulation of unbalanced optimal transport [165], the constraint concerning the preservation of marginal distributions is weakened as the following formulation:

$$\mathbf{L}^{\tau}_{\mathbf{C}}(\mathbf{a},\mathbf{b}) = \min_{\tilde{\mathbf{a}},\tilde{\mathbf{b}}} \mathbf{L}_{\mathbf{C}}(\mathbf{a},\mathbf{b}) + \tau_1 \mathbf{D}_{\varphi}(\mathbf{a},\tilde{\mathbf{a}}) + \tau_2 \mathbf{D}_{\varphi}(\mathbf{b},\tilde{\mathbf{b}}) \tag{4.20}$$

$$= \min_{\mathbf{P}\in\mathbb{R}^{n\times m}_{+}} \langle \mathbf{C},\mathbf{P}\rangle + \tau_1 \mathbf{D}_{\varphi}(\mathbf{P}\mathbb{1}_m|\mathbf{a}) + \tau_2 \mathbf{D}_{\varphi}(\mathbf{P}^T\mathbb{1}_n|\mathbf{b}) \tag{4.21}$$

where $\mathbf{D}_{\varphi}$ includes ℓ^2 norm, ℓ^1 norm and Kulback–Leibler divergence $\mathbf{D}_{KL}$. The divergences D_{φ} in (4.21) penalize the deviation of the marginal distribution from the distributions α and β, and then it reinforces the mass conservation constraint (4.15). The cost $\mathbf{L}^{\tau}_{\mathbf{C}}(\mathbf{a},\mathbf{b})$ is regulated by a factor τ. On the one hand, by adjusting τ at a moderate value, the constraint (4.15) is lightened, and eventually, the mass conservation condition is not held. In other words, the mass being transferred from the *source* to *target* is no longer equal to one in Unbalanced Optimal Transport. On the other hand, in the case limit when $\tau_1 = \tau_2 \to +\infty$, the unbalanced optimal transport problem recovers its original form (4.14) (i.e., the balanced case).

As mentioned previously, in our overlapping multi-camera scenario, collaboratively tracking between cameras requires an online target matching in every pair of cameras. The collaboration mechanism ensures that if a target is out of sight of one camera, it is still being tracked by another. As a result, this helps the recovery procedure more robust when the target reappears again on the original camera's view. It is a frequent case that some targets are being seen by one, but not the other, for several reasons, such as each camera has its own observation zone (apart from the common overlapping zone between every pair of cameras) or target being hidden by an obstacle in one view. This can be implied that not all target found in one view have their correspondence in another view. Consequently, the Unbalanced Optimal Transport (UOT) can adequately handle this target matching problem. In computational practice, the Unbalanced Optimal Transport problem is resolved by a generalized version of the Sinkhorn's algorithm [44].

After reformulating the target association across cameras as an unbalanced optimal transport problem, there is a challenging issue related to the feature space for Optimal Transport. Indeed, the target's appearance features Φ, including raw image patch or extracted from frame image via deep neural nets are bounded, e.g., image patch's values are between 0 and 1. Meanwhile, the position of targets x is not bounded in the tracking scene with multiple cameras. Unlike the single-camera case where the image size can limit the target's feasible position, online tracking with multiple cameras requires a common space, e.g., the ground, where the position of targets can be converted into the same measure unit on a measure space, which is not bounded. Furthermore, the combination of these two features, which is used to compute the distance for Optimal Transport, demands fairness and balance. Therefore, in the next section, we suggest a novel approach based on deep neural networks to learn the distance between the *source* and *target* distributions.

4.3.3 Ground cost learning for UOT-based targets association across cameras

This section describes our proposed deep distance learning framework, which helps to compute an appropriate distance for the Optimal Transport problem between targets of one camera and those of another. More concretely, on each camera, the appearance feature of each target is extracted from its image patch via a deep convolutional neural network, e.g. VGG [193], ResNet [99], Inception [196]. Meanwhile, its position x is determined by projecting the target's feet point on the image into the ground plane via the homography matrix of the camera. Both the appearance and location feature vectors are the input of a deep neural network whose output is a unique point in a feature space $\mathcal{X}$. The collection of all points mapped from all targets of a camera via the deep neural net makes up a distribution. As discussed in the previous section, given two distributions created from targets of a pair of cameras, target association in a pair of two cameras is an Unbalanced Optimal Transport from a distribution, called *source*, to the other one, called *target*. Therefore, the Optimal Transport plan is followed by a thresholding step, to obtain a binary matrix as the association matrix of targets between the two cameras. Fig. 4.14 displays the pipeline of our distance learning framework, which aims to learn ground cost between targets across cameras so that the optimal transport plan approximates the ground-truth assignment.

Because our deep-learning-based method is a supervised learning approach, it is required a training data with labels. Our training data are directly generated from the training sequences of a dataset. Precisely, for each frame of videos, every pair of cameras gives a single assignment as the label of a sample, while the data of the sample is extracted from the ground-truth bounding boxes via the deep extracting feature net. In the case of N cameras in the network, the combination of camera possible pairs is $N(N-1)/2$, which is also the number of samples generated in each frame instant.

In our formulation, for each target i, given $\Phi_i \in \mathbb{R}^{2048}$ (i.e., output of ResNet50 backbone [99]) and $x_i \in \mathbb{R}^2$ (i.e., target coordinate on ground), the embedding function f_w, via our deep neural network (see Fig. 4.15), projects the appearance feature and location of target i into the feature space $\mathcal{X}$,

$$f_w : (\Phi, x) \to \mathcal{X},$$

where w is the parameters of the deep neural net. As a result of unbalanced optimal transport, the transport plan shows the mass flows from point i of *source* to point j of *target*. Based on the properties of optimal transport [165], any pair of close points distributions *source* and *target* results in a significant mass flow on its transport plan compared with others. Fig. 4.16 (a) is an optimal transport plan in which i^{th} row represents the mass of the i^{th} *source* point being transferred to all *target*. Since only consistent mass transfers from one point on *source* to an unique point in *target* is sought, the optimal transport plan between *source* and *target* is expected to be well "sparse", which means that the matching can be deduced straightforwardly (see Fig. 4.16 (b)) by thresholding the optimal transport plan. We then can obtain the assignment from *source* to *target*.

In terms of optimization, the dissimilarity between the optimal transport plan $\mathbf{P}_\varepsilon(\alpha, \beta)$ and the ground-truth assignment $\mathbf{G}(\alpha, \beta)$ is measured by a loss function $\mathcal{L}$. The learnable parameters w of our neural net is then determined via a minimization problem:

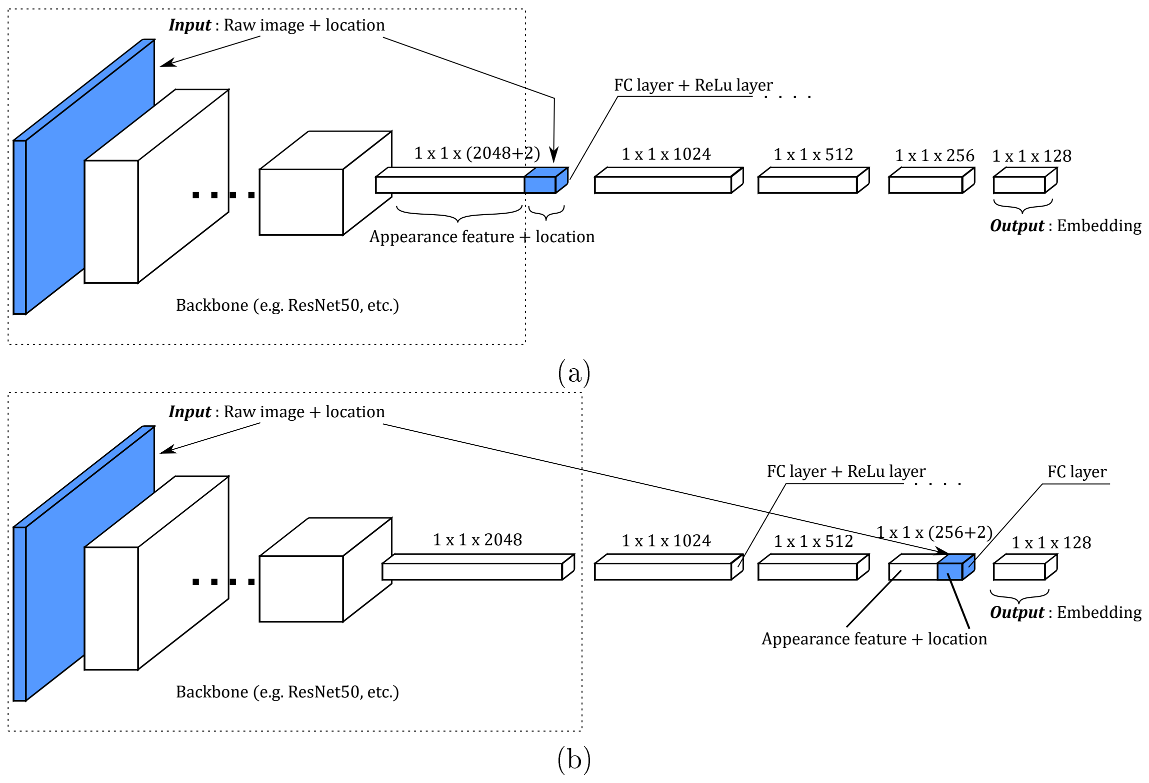


Figure 4.15 – Proposed distance learning neural net. The neural net consists of a CNN backbone (e.g. ResNet50 in our case), which extract appearance features from raw image, and a series of Fully Connected (FC) layers with ReLU layers as activations. Model (a) with locations at the bottom of the deep distance network, meanwhile, model (b) with locations at the second last FC layer.

$$w = \arg\min_{w} \mathcal{L}\left(\mathbf{P}_{\varepsilon}\left(.;w\right), \mathbf{G}(.)\right) \tag{4.22}$$

The loss functions in our framework are formulated as following. Given two sets of targets $\left\{v_i^{k_1}\right\}_n$ and $\left\{v_j^{k_2}\right\}_m$, each belongs to a single camera, the assignment task is to find the correspondence of common targets in the pair of cameras k_1 and k_2 while excluding the targets which can be seen in only one view. Given $\mathbf{P}_\varepsilon \in \mathbb{R}_+^{n\times m}$ the transport plan and $\mathbf{G} \in \{0,1\}^{n\times m}$ the ground-truth assignment, we propose our loss which is delivered from the dual problem of regularized optimal transport problem (4.18). Therefore, the first order condition to reach the optimal solution [165] yields to:

$$\frac{\partial \Theta(\mathbf{P}, \mathbf{f}, \mathbf{g})}{\partial \mathbf{P}_{ij}} = 0 \tag{4.23}$$

where $\Theta(\mathbf{P}, \mathbf{f}, \mathbf{g}) = \langle \mathbf{C}, \mathbf{P}\rangle - \varepsilon \mathbf{H}(\mathbf{P}) + \langle \mathbf{a} - \mathbf{P}\mathbb{1}_m, \mathbf{f}\rangle + \langle \mathbf{b} - \mathbf{P}^T\mathbb{1}_n, \mathbf{g}\rangle$

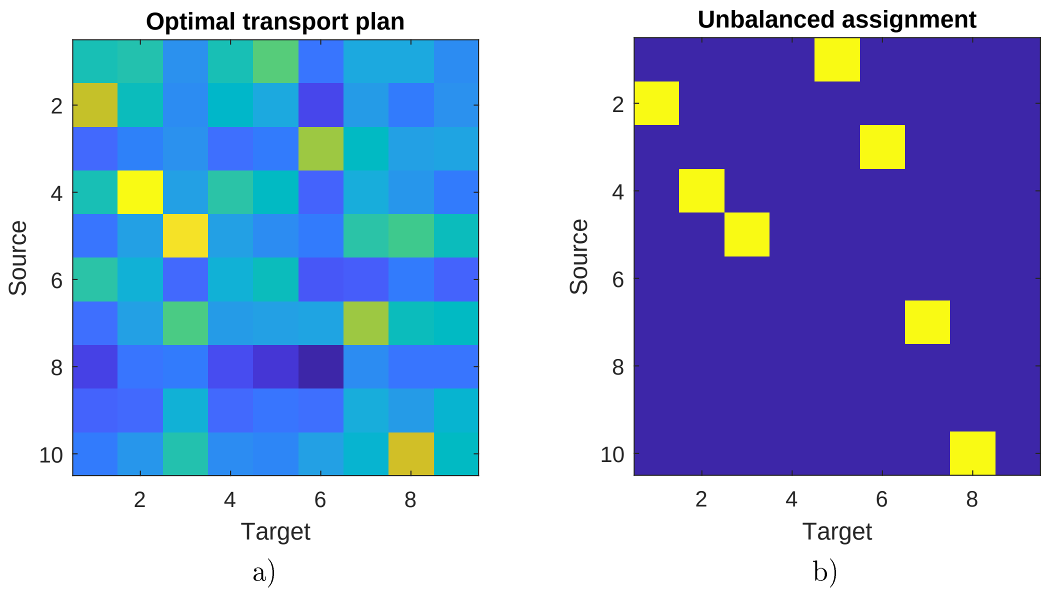


Figure 4.16 – Comparison between optimal transport plan (a) and assignment matrix (b)

$$\Rightarrow \quad \varepsilon \log(\mathbf{P}_{ij}) + \mathbf{C}_{ij} - \mathbf{f}_i - \mathbf{g}_j = 0 \tag{4.24}$$

$$\Rightarrow \quad \log(\mathbf{P}_{ij}) = \frac{\mathbf{f}_i + \mathbf{g}_j - \mathbf{C}_{ij}}{\varepsilon} \tag{4.25}$$

In the training phase of our experiments, by default, both total masses of the measure α and β are equal 1, which means the mass is distributed uniformly via the same-mass Diracs of the distributions, i.e., $\mathbf{a}_i = 1/n$ and $\mathbf{b}_j = 1/m$. The constraint of mass conservation in the expression (4.15) leads to the sum of all elements of the optimal transport plan $\mathbf{P}$ smaller or equal 1. The equality happens in the case of the balanced optimal transport, and the inequality for the unbalanced optimal transport case. As a result, the ground-truth assignment $\mathbf{G}$ needs to be normalized to keep the assignment matrix and optimal transport plan comparable. Hence, from the expression of the transport plan (4.25), our loss is formulated as follows:

$$\begin{aligned}
\mathcal{L}(\mathbf{P}_\varepsilon, \mathbf{G}) &= \sum_{i,j} \left| \log \left(\frac{\mathbf{G}'_{ij}}{\max(\mathbf{P}_{ij}, \sigma)} \right) \right| \\
&= \sum_{i,j} \left| \log(\mathbf{G}'_{ij}) - \max(\log(\mathbf{P}_{ij}), \log \sigma) \right| \\
&= \sum_{i,j} \left| \log(\mathbf{G}'_{ij}) - \max \left(\frac{\mathbf{f}_i + \mathbf{g}_j - \mathbf{C}_{ij}}{\varepsilon}, \log \sigma \right) \right| \qquad (4.26)
\end{aligned}$$

where the normalized assignment coupling $\mathbf{G}_{i,j}$ is defined as

$$\mathbf{G}'_{ij} = \frac{\mathbf{G}_{ij}}{\sum_{i,j} \mathbf{G}_{ij} + \gamma} + \sigma \geq \sigma \tag{4.27}$$

with σ is a tiny threshold value, and γ is a normalization constant. This threshold value is added to avoid the logarithm of zero value in the loss function (4.26) and to set a margin for near-zero transport, which does not contribute to the distance loss if its value is extremely low. Additionally, the parameters of our neural net w are updated iteratively

```
   Input  : N source-target pairs $\big\{\{\Phi_i^1, x_i^1\}_n, \{\Phi_i^2, x_i^2\}_m\big\}_N$;
            Ground-truth assignment $\{G_i\}_N$
   Output: Optimized parameters $w$
 1 Initializing net parameters $w$
 2 // optimization loop
 3 foreach epoch do
 4     foreach source-target pair do
 5         // Embedding targets to feature space
 6         $\{X_i\}_n \leftarrow f\left(\{\Phi_i^1, x_i^1\}_n, w\right)$
 7         $\{Y_i\}_m \leftarrow f\left(\{\Phi_i^2, x_i^2\}_m, w\right)$
 8         Computing Optimal Transport plan $P_\varepsilon(\{X_i\}_n, \{Y_i\}_m)$ via Eq. (4.19)
 9         Computing loss $\mathcal{L}(P_\varepsilon, G)$;
10         Back-propagation;
11         Updating parameters $w$;
12     end
13 end
```

Algorithm 6: Deep distance learning algorithm.

via minimizing the loss $\mathcal{L}$. The derivation of the loss function to the net parameters $\partial\mathcal{L}/\partial w$ is computed via back-propagation, which occurs after every optimal transport of a *source-target* pair from two cameras. All distance learning steps are resumed in Alg. 6.

4.3.4 Performance evaluation

4.3.4.1 Implementations

In our implementations, we build two versions of distance learning neural nets in order to compute the source-target distance in the Optimal Transport:

(a) The appearance feature of targets obtained from the backbone of ResNet50, in addition to their location, is considered as the inputs of our distance learning network. Our deep network is a series of Full Connected Layers with ReLU layer on the top of each. The outputs of FC layers are respectively 1024, 512, 256 and 128 (see Fig. 4.15 (a)).

(b) The second model is modified from the original one, but instead of using locations in the first layer, it is concatenated with the second last output layer. The intuition behind is to emphasize the location feature of targets, because, in tracking applications, positions of targets are crucial to the performance of tracking algorithm (see Fig. 4.15 (b)).

In the deployment phase, numerous experiments with different configurations are conducted within the framework of multiple camera tracking of the paper [132]. Therefore, the target assignment or matching is applied on only two cameras, collaborative tracking in our multiple camera approach occurs on pairs of cameras, but one camera can reach all others through the whole tracking process. Precisely, at each frame, each camera consecutively pairs with all other cameras, then within each pair of cameras, an optimal transport plan $\mathbf{C}$ is computed in order to link targets from one camera to the other in the pair. As mentioned in the section 4.3.2, the value of each cell C_{ij} of the optimal transport plan implies how likely element i of *source* set is matched to element j of *target* set based on the amount of mass being transferred, named *OT value*. Therefore, each missing target on one view is associated with its corresponding target on each other view by an OT value obtained from its optimal transport plan. Hence, the tracking result of the missing target is replaced by the "tracked" target with the *highest* value among its correspondences on all other cameras.

The Optimal Transport algorithm we used in this paper is a public Optimal Transport library [80] on Python with GPU parallelization support, named KeOps-GeomLoss[4]. The parameters of the unbalanced optimal transport problem (4.18) are set as follows:

- $p = 2$
- "blur" $= 0.5 \rightarrow \varepsilon = blur^p$
- "reach" $= 0.1 \rightarrow \tau_1 = \tau_2 = \tau = reach^p$
- $D_\varphi = KL$: soft Kulback–Leibler divergence

Meanwhile, the other parameters in the expression (4.27) are adjusted for $\sigma = 10^{-8}$ and $\gamma = 10^{-4}$. The detailed implementation of our deep distance learning method will be available publicly on our project page.

4.3.4.2 Tracking benchmark

Datasets. For a consistent comparison with our association approach in the section 4.2, the datasets used in our experiments remains the same, including the PETS2009 [78] and EPFL Multi-camera Pedestrian Videos [20] datasets. However, the sequence "PETS09-S2L1" with 7 cameras, we add another sequence with only 3 cameras "PETS09-S2L2" for our performance benchmark. Indeed, the additional scenario of surveillance is to track an influx of people moving on the roads with different speed, and this makes it far more crowded that "PETS09-S2L1". Since the lack of cameras in this sequence, we set up dual-camera tracking experiments on view 1 and view 2. View 3 is excluded, due to its small impact on the sequence and the absence of ground-truth data as well.

Detection. In all Tracking-by-Detection approaches, the detector plays an important role in tracking performance. Therefore, in these experiments, detections in video frames are generated by the public high-accuracy detectors, including OpenPose [36] and Mask R-CNN [97].

[4]https://www.kernel-operations.io/geomloss/

Evaluation metric. To evaluate the performance of our second data association approach in our multi-camera tracking framework, we adopt the same metric used in the section 4.2.4, which includes the CLEAR MOT and ID measures scores such as: MOTA (multiple-object tracking accuracy), MOTP (multiple-object tracking precision), IDs (identity switches), IDF1 (ID F1-score), IDP (ID precision), IDR (ID Recall), False Positive (FP) and False Negative (FN).

4.3.4.3 Experimental results

The results shown in the following tables are the average values of all views. Concretely, the overall tracking results of the PETS sequence can be seen in the Table 4.7, Table 4.8 and Table 4.9. Each score column has either a ↑ or a ↓ indicating whether better corresponds to higher or lower, respectively. The **bold** indicates the best scores, and the ***bold and italic*** for the second best. We regroup all methods in our experiments into three categories: the first group single-camera methods [229] as the baseline, full multi-camera methods which operate with all cameras in the second group, and dual-camera methods in the last group. Consequently, the experiments on the dual-camera sequence "PETS09-S2L2" only display the results of the single-camera method (as the baseline) and dual-camera methods in Tab. 4.9. For convenience, we denote different features for the affinity measure of targets across cameras, which are used in our multi-camera methods, as path (i.e., trajectory), pos (i.e., position) and DL (a) or (b) (i.e., distance learning model (a) or (b)).

Method + Feature	**IDF1↑**	**IDP ↑**	**IDs↓**	**MOTA↑**	**MOTP↑**
Single cam [229]+∅	57.49	62.24	333	68.44	68.83
All cam [132]+path	**72.8**	**78.53**	**98**	***73.26***	***70.69***
KSP [81]+∅	21.51	18.16	812	-29.63	64.27
Dual-cam [132]+path	67.96	72.72	***126***	***73.4***	70.65
UOT+DL(a)	***68.15***	***73.41***	153	73.16	**70.81**
UOT+DL(b)	66.71	71.73	174	72.19	70.65
UOT+pos	66.04	70.66	163	72.76	70.60

Table 4.7 – Scores on "PETS09-S2L1" multi-camera sequence (with OpenPose detector [36]).

Primarily, our multi-camera tracking method aims to address hard occlusion problems. It leads to an important reduction of identity switches and a significant improvement of ID measures in comparison with the single-camera method. In the sequence "PETS09-S2L1", the targets have their complex movements and mutual interactions inside the overlapped area of the tracking scene. All the methods using the target trajectory as features of affinity measure show the better scores in all categories, in comparison with the approaches, which do not consider historical position record of targets (i.e., trajectories), but only the instant measure including image patch and position of targets. In detail, the method with full camera collaboration (All-cam) [132] shows off its superiority. Meanwhile, our Unbalanced Optimal Transport approach (UOT) is less robust, but still significantly improves tracking scores compared to single-camera approach. Notwithstanding, in the tests with the

Method + Feature	**IDF1↑**	**IDP ↑**	**IDs↓**	**MOTA↑**	**MOTP↑**
Single cam [229]+∅	21.88	25.66	388	55.98	72.53
All cam [132]+path	21.32	25.05	461	54.14	72.47
KSP [81]+∅	**25.85**	23.51	695	19.57	62.26
Dual-cam [132]+path	23.23	26.72	***382***	***56.71***	72.43
UOT+DL(a)	24.36	**31.40**	**305**	46.86	**72.91**
UOT+DL(b)	***25.15***	***28.88***	385	**56.93**	**72.63**
UOT+pos	22.00	25.32	381	56.40	72.48

Table 4.8 – Scores on “terrace1” multi-camera sequence (with OpenPose detector [36]).

Method + Feature	**IDF1↑**	**IDP ↑**	**IDs↓**	**MOTA↑**	**MOTP↑**
Single cam [229]+∅	53.46	55.53	321	63.66	***75.14***
Dual-cam [132]+path	***55.63***	***57.67***	327	**63.79**	**75.16**
UOT+DL(a)	53.16	55.76	**310**	62.62	**75.16**
UOT+DL(b)	53.75	55.83	329	63.51	**75.16**
UOT+pos	**57.30**	**59.38**	***312***	***63.74***	74.98

Table 4.9 – Scores on “PETS09-S2L2” dual-camera sequence (with Mask R-CNN detector [97]).

sequence “EPFL/terrace1”, the tracking scene composes 8 identities moving mainly around a relatively small area covered by a smaller camera number, which makes the scene more crowded and targets hardly seen by all cameras. Consequently, the original approach [132] failed to improve tracking results, because, with a smaller camera amount, it is obviously less probable that there are more than 2 or 3 cameras observed the same target at the same time. The results in Tab. 4.8 show that all other approaches with dual-camera mode perform significantly better than the original ones. The next remark is that in the scenario where there are only short trajectories that can be seen, the trajectory feature is less reliable. In other words, shorter trajectories, less effective the original approach is. Hence, in Tab. 4.8, our distance learning method based on Optimal Transport outweighs the conventional approaches which only use position or trajectory of target as input feature for affinity measure.

Finally, on the tests with dual-camera sequence “PETS09-S2L2”, we excluded the methods which require more than 2 cameras to be operational, including all camera [132] and KSP [20]. As single object trackers can generate long trajectories for targets, trajectories are still an important feature to distinguish targets that we can see in Table 4.9. The approach [132] with dual-camera only using trajectory as target features archived the second-best result on ID-measures and the best on MOT-scores. Meanwhile, our UOT dual-cam approach based on position only obtained the best scores on ID-measures and the second-best on MOT-scores. Unfortunately, two of our UOT methods using distance learning could not outperform others in this sequence. The detailed results of all experiments are shown in Appendix A.

Chapter 5

Conclusion and perspectives

In this dissertation, we contributed to several aspects of Multiple Camera Multiple Object Tracking problems that push the advances in the field forwards. The following part summarizes all our research work during this Ph.D.

The Ph.D. program began in late 2016 and aimed to resolve problems in detection and tracking multi-target with multiple cameras towards an intelligent camera system in urban environments. We directed our research work to online multiple camera tracking applications in the context of frame-synchronized, calibrated, and overlapping cameras systems. Our first contribution relates to developing a particle-based multi-cameras framework to enable multiple cameras to collaborate in order to tackle occlusion problems. Following the multiple target tracking schemes in the literature on remote sensors, we proposed a particle generating step on the ground plane to mitigate the degeneration of the probabilistic model through time frames and avoid the error propagation across cameras through the collaboration process. We proposed the use of sparse coding to encode the target's appearance in a dictionary, which is well-adapted in our proposed multi-camera framework. The obtained results showed the effectiveness of using multiple cameras to handle long-time occlusion in comparison with single view single object tracking methods. Unfortunately, due to the instability of the sparse-coding-based appearance model combining with a probabilistic model in multiple object tracking, our framework could not reach the state-of-the-art performance in multiple object tracking. Therefore, we redirected our research toward developing non-probabilistic tracking algorithms, which is adequately fitted for visual tracking with multiple cameras.

Our second contribution concerns extending a single view MDP tracking approach in a multiple camera framework. Indeed, our novel proposed framework allows a camera in the network to share its own tracking results with others. Moreover, multiple camera tracking involves how to identify targets across cameras, then our work focuses on differentiating targets based on their physical appearance and trajectory. We have evaluated the impact of different features on discriminating targets across different views with tracking results. Our experiments demonstrate the superiority of our multiple camera approach to single-camera ones in multiple objects tracking tasks. The experimental results also imply that targets' trajectory is a strong discriminative feature among other appearance features in multiple camera tracking. One of the setbacks of this framework is that the physical appearance of targets is not fully employed to distinguish targets in multi-camera tracking problems. Consequently, this led our latter research to combine the appearance and trajectories into a more robust and effective discriminator of targets in multiple camera's views.

Our final contribution involves the target association problem in pairs of cameras. As an attempt to properly combine two robust discriminative features of targets, we introduced a novel approach to associate targets in one camera to those in another. The original target association is an unbalanced assignment between two unequal sets, which is reformulated as the Unbalanced Optimal Transport problem. As a result of the Optimal Transport plan, the target assignment would be deduced, which helps identify targets of one camera among targets of another camera during the tracking process. We adapted the target-matching process of camera pair to our dual-camera tracking framework in the context of multiple camera tracking problems. The experimental results revealed a significant improvement of our dual-camera approach compared with full camera collaboration one in the scenario, where the targets' trajectory is being cut short and segmented. Our Optimal Transport approach in dual-camera tracking shows the effectiveness and robustness in tracking with multiple camera configurations. Our method could be applied for a generic multi-camera tracking case in which overlapping condition for all cameras is not necessarily held.

Our future work on multiple camera tracking relates to controlling and automating multi-camera systems. This makes collaborative tracking with multiple cameras achievable and practical in both cases of static cameras, in which switching cameras to track targets is automated, and dynamic cameras, in which they can operate some actions such as panning, tilting and zooming themselves. For the last recent years, reinforcement learning has attracted attention of researchers in automation and robotics domains. Indeed, reinforcement learning is one of three main branches of machine learning, alongside with supervised learning and unsupervised learning, which concerns with how software agents should take actions in a predefined environment in order to maximize the cumulative award received. Most recent papers on reinforcement learning applied in robotics applications have shown impressive and promising results in future technologies such as autonomous driving, robot-assisted surgery, etc. Therefore, automation by reinforcement learning should be explored in video surveillance technologies, particularly, in multiple camera tracking systems.

Annexes

Appendix A

Detail of experimental results

We mention in the appendix the description of several important indexes of MOT metric in the table below (for the full description, please visit motchallenge.net).

Index	Better	Perfect	Description
IDF1	higher	100%	ID F1 Score [178]. The ratio of correctly identified detections over the average number of ground-truth and computed detections.
MOTA	higher	100%	Multiple Object Tracking Accuracy [22]. This measure combines three error sources: false positives, missed targets and identity switches.
MOTP	higher	100%	Multiple Object Tracking Precision [22]. The misalignment between the annotated and the predicted bounding boxes.
MT	higher	100%	Mostly tracked targets. The ratio of ground-truth trajectories that are covered by a track hypothesis for at least 80% of their respective life span.
ML	lower	0%	Mostly lost targets. The ratio of ground-truth trajectories that are covered by a track hypothesis for at most 20% of their respective life span.
FP	lower	0	The total number of false positives.
FN	lower	0	The total number of false negatives (missed targets).
IDs	lower	0	The total number of identity switches. Please note that we follow the stricter definition of identity switches as described in [141].

Please see the full tables of experiments in the next pages.

Table of experimental results on sequence **PETS09-S2L1** in the PETS2009 dataset

Original MDP method

Sequence	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View1	57.8	59.6	56.1	85.8	91.2	0.47	19	12	7	0	371	635	95	106	75.4	72.2	77.5
View5	47.6	56.1	41.4	67.7	91.7	0.21	28	8	19	1	167	884	71	105	58.9	71	61.5
View6	50	57.2	44.4	71.3	91.8	0.22	29	11	17	1	171	777	77	75	62.1	68.1	64.9
View7	53.2	60.3	47.6	73.1	92.6	0.22	26	11	13	2	172	793	74	81	64.8	65.5	67.2
View8	69.1	69.7	68.5	86.4	87.9	0.47	29	18	11	0	374	428	35	53	73.5	65.5	74.5
Overall	56.5	61.1	52.5	78.0	90.9	0.32	131	60	67	4	1255	3514	352	420	68.0	68.8	70.2

Our MDP multi-view method

Sequence	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View1	64.8	66.4	63.2	86.8	91.3	0.47	19	14	5	0	372	592	68	106	76.9	72.3	78.4
View5	62.2	73.8	53.7	67.8	93	0.17	28	6	22	0	138	878	52	131	60.8	71.1	62.7
View6	63.9	71.9	57.5	73.6	92	0.22	29	14	14	1	174	713	52	81	65.3	68.4	67.1
View7	65.7	74.6	58.7	73.8	93.7	0.18	26	13	11	2	146	773	33	87	67.7	65.5	68.8
View8	74.5	74.9	74.2	86.6	87.4	0.49	29	19	10	0	393	423	35	58	73	65.7	74.1
Overall	66.4	71.6	62.0	78.9	91.2	0.31	131	66	62	3	1223	3379	240	463	69.8	68.9	71.2

Overall scores of both algorithm in all views

Method	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
MDP	56.5	61.1	52.5	78.0	90.9	0.32	131	60	67	4	1255	3514	352	420	68.0	68.8	70.2
Our MDP	66.4	71.6	62.0	78.9	91.2	0.31	131	66	62	3	1223	3379	240	463	69.8	68.9	71.2
Tolerate	9.9	10.5	9.5	0.9	1.1	-0.01	0	6	-5	-1	-32	-135	-112	43	1.8	0.1	1.0
Gain (%)	17.5	17.2	18.1	1.2	1.2	-3.1	0	10.0	-7.5	-25	-2.6	-3.8	-31.8	10.2	2.7	0.2	1.4

Scores of both algorithm in the main view (view 1)

Method	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
MDP	57.8	59.6	56.1	85.8	91.2	0.47	19	12	7	0	371	635	95	106	75.4	72.2	77.5
Our MDP	64.8	66.4	63.2	86.8	91.3	0.47	19	14	5	0	372	592	68	106	76.9	72.3	78.4
Tolerate	7	6.8	7.1	1.0	0.1	0	0	2	-2	0	1	-43	-27	0	1.5	0.1	0.9
Gain (%)	9.9	10.5	12.7	1.2	0.1	0	0	16.7	-28.6	0	-0.3	-6.8	28.4	0	2.0	0.1	1.2

Table of experimental results on sequence **Terrace1** in the EPFL dataset

Original MDP method

Sequence	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
terrace1-c0	8.2	13.7	5.8	39.4	92.3	0.19	9	0	9	0	194	3574	189	171	32.9	72	36.1
terrace1-c1	10.9	17.5	7.9	41.2	91.7	0.24	9	0	9	0	245	3876	170	181	34.9	75.1	37.5
terrace1-c2	8.5	15.6	5.9	34.9	92.4	0.18	9	0	9	0	184	4168	170	170	29.4	71.0	32.0
terrace1-c3	12.2	18.6	9.1	46.5	95.5	0.10	9	0	9	0	104	2518	145	135	41.2	72.0	44.3
Overall	9.9	16.4	7.1	40.1	92.9	0.18	36	0	36	0	727	14136	674	657	34.2	72.7	37.0

Our MDP multi-view method

Sequence	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
terrace1-c0	13.3	22.1	9.5	39.1	91.3	0.22	9	0	9	0	219	3593	190	203	32.1	72.1	35.3
terrace1-c1	11.2	18.4	8.1	40.1	91.6	0.24	9	0	9	0	243	3953	187	201	33.5	75.1	36.3
terrace1-c2	10.2	18.6	7.0	34.7	92.6	0.18	9	0	9	0	177	4183	168	200	29.3	70.9	31.9
terrace1-c3	15.4	23.7	11.4	45.9	95.5	0.10	9	0	9	0	102	2549	144	153	40.6	71.7	43.7
Overall	12.3	20.6	8.8	39.5	92.6	0.19	36	0	36	0	741	14278	689	757	33.5	72.6	36.4

Overall scores of both algorithm in all views

Method	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
MDP	9.9	16.4	7.1	40.1	92.9	0.18	36	0	36	0	727	14136	674	657	34.2	72.7	37.0
Our MDP	12.3	20.6	8.8	39.5	92.6	0.19	36	0	36	0	741	14278	689	757	33.5	72.6	36.4
Tolerate	2.4	4.2	1.7	-0.6	-0.3	0.01	0	0	0	0	14	142	15	100	-0.7	-0,1	-0.6
Gain(%)	24.2	25.6	23.9	-1.5	-0.3	5.6	0	0	0	0	1.9	1.0	2.2	15.2	-2.0	-0.1	-1.6

Table of experimental results on both sequence (Heuristics path)

Sequence PETS09-S2L1

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View1	69.93	70.61	69.26	91.09	92.86	0.395	18	15	3	0	235	299	27	75	83.27	70.71	84.04
View5	62.77	71.42	55.99	69.56	88.78	0.27	22	9	13	0	155	534	37	92	58.67	68.47	60.628
View6	73.81	79.03	69.24	81.39	92.90	0.18	23	17	6	0	105	314	18	28	74.10	72.94	75.09
View7	56.92	67.70	49.10	69.04	95.21	0.13	20	10	8	2	77	686	27	38	64.35	69.38	65.50
View8	74.36	76.39	72.43	87.21	91.98	0.28	23	18	5	0	164	276	17	28	78.82	71.31	79.55
Overall	67.96	72.72	63.78	81.12	92.49	0.25	106	69	35	2	736	2109	126	261	73.40	70.65	74.51

Sequence EPFL-terrace1

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
terrace1-c0	28.10	33.64	24.13	63.53	88.58	0.43	9	0	9	0	310	1380	88	147	53.01	72.92	55.29
terrace1-c1	20.44	23.11	18.33	69.54	87.71	0.52	9	0	9	0	378	1182	130	184	56.44	73.09	59.74
terrace1-c2	21.78	24.87	19.37	69.83	89.67	0.42	9	2	7	0	305	1143	88	151	59.46	71.79	61.73
terrace1-c3	22.80	25.86	20.38	69.59	88.30	0.43	9	2	7	0	308	1016	76	122	58.10	71.89	60.32
Overall	23.23	26.72	20.54	68.09	88.56	0.45	36	4	32	0	1301	4721	382	604	56.71	72.43	59.28

Table of experimental results on both sequence (UOT-pos)

Sequence PETS09-S2L1

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View1	75.16	76.14	74.21	90.82	93.18	0.37	18	15	3	0	223	308	35	81	83.13	70.64	84.12
View5	62.91	71.78	55.99	69.38	88.96	0.25	22	7	15	0	151	537	30	95	59.07	68.18	60.69
View6	68.92	73.68	64.73	81.15	92.38	0.19	23	17	6	0	113	318	29	30	72.73	72.72	74.36
View7	49.70	58.21	43.37	70.04	94.00	0.17	20	11	7	2	99	664	47	37	63.45	69.57	65.49
View8	66.86	69.02	64.83	86.15	91.71	0.28	23	17	6	0	168	299	22	26	77.34	71.40	78.30
Overall	66.04	70.66	61.98	80.97	92.30	0.25	106	67	37	2	754	2126	163	269	72.76	70.60	74.20

Sequence EPFL-terrace1

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
terrace1-c0	26.31	31.08	22.81	64.30	87.61	0.48	9	0	9	0	344	1351	92	140	52.78	73.13	55.15
terrace1-c1	20.08	22.73	17.99	69.64	87.98	0.51	9	0	9	0	369	1178	123	173	56.96	72.75	60.08
terrace1-c2	20.57	23.63	18.21	68.62	89.04	0.44	9	1	8	0	320	1189	94	155	57.69	71.99	60.12
terrace1-c3	21.10	24.14	18.74	69.08	89.01	0.40	9	1	8	0	285	1033	72	121	58.40	72.02	60.50
Overall	22.00	25.32	19.45	67.89	88.40	0.46	36	2	34	0	1318	4751	381	589	56.40	72.48	58.96

Table of experimental results on both sequence (UOT-model (a))

Sequence PETS09-S2L1

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View1	78.26	79.84	76.74	90.01	93.64	0.35	18	15	3	0	205	335	33	78	82.92	70.66	83.85
View5	61.32	70.85	54.05	69.10	90.58	0.21	22	9	13	0	126	542	36	88	59.86	68.72	61.83
View6	75.83	81.81	70.66	80.50	93.21	0.17	23	16	7	0	99	329	23	25	73.27	73.25	74.55
View7	49.12	57.74	42.74	70.13	94.76	0.15	20	11	6	3	86	662	41	35	64.40	69.84	66.17
View8	69.08	71.49	66.82	86.05	92.07	0.27	23	17	6	0	160	301	20	28	77.71	71.44	78.58
Overall	68.15	73.41	63.60	80.58	93.01	0.23	106	68	35	3	676	2169	153	254	73.16	70.81	74.51

Sequence EPFL-terrace1

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
terrace1-c0	28.05	35.52	23.18	57.77	88.54	0.40	9	0	9	0	283	1598	90	140	47.91	73.28	50.24
terrace1-c1	23.57	26.99	20.93	68.17	87.90	0.50	9	0	9	0	364	1235	100	166	56.21	73.19	58.74
terrace1-c2	19.34	28.34	14.67	45.66	88.18	0.32	9	0	8	1	232	2059	63	98	37.87	72.32	39.49
terrace1-c3	26.49	36.11	20.92	52.26	90.19	0.26	9	0	9	0	190	1595	52	91	45.02	72.60	46.52
Overall	24.36	31.40	19.90	56.15	88.60	0.37	36	0	35	1	1069	6487	305	495	46.86	72.91	48.91

Table of experimental results on both sequence (UOT-model (b))

Sequence PETS09-S2L1

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View1	77.01	78.53	75.55	90.25	93.80	0.34	18	15	3	0	200	327	35	75	83.24	70.66	84.24
View5	61.51	71.08	54.22	68.59	89.91	0.23	22	8	14	0	135	551	41	92	58.55	68.35	60.80
View6	70.10	74.47	66.21	81.27	91.40	0.22	23	17	6	0	129	316	38	36	71.37	72.87	73.53
View7	48.76	58.16	41.97	68.01	94.25	0.15	20	9	9	2	92	709	39	41	62.09	69.53	63.78
View8	68.13	70.04	66.31	86.38	91.24	0.30	23	18	5	0	179	294	21	28	77.11	71.38	78.02
Overall	66.71	71.73	62.34	80.33	92.43	0.25	106	67	37	2	735	2197	174	272	72.19	70.65	73.73

Sequence EPFL-terrace1

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
terrace1-c0	30.40	36.18	26.22	63.87	88.15	0.46	9	0	9	0	325	1367	87	152	52.99	73.20	55.23
terrace1-c1	24.06	27.16	21.60	69.33	87.17	0.55	9	0	9	0	396	1190	118	184	56.08	73.33	59.07
terrace1-c2	22.45	25.36	20.14	71.34	89.83	0.42	9	3	6	0	306	1086	107	134	60.44	72.24	63.21
terrace1-c3	23.76	27.29	21.04	68.84	89.29	0.38	9	1	8	0	276	1041	73	118	58.40	71.65	60.53
Overall	25.15	28.88	22.28	68.34	88.58	0.45	36	4	32	0	1303	4684	385	588	56.93	72.63	59.51

Table of experimental results on Sequence PETS09-S2L2 (dual camera tracking)

Single-camera MDP

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View 1	54.80	59.97	50.45	77.85	92.53	1.42	68	32	36	0	617	2175	167	264	69.86	74.85	71.54
View 2	52.03	51.28	52.80	80.60	78.28	4.31	66	44	22	0	1877	1628	154	211	56.40	75.47	58.21
Overall	53.46	55.53	51.54	79.12	85.24	2.86	134	76	58	0	2494	3803	321	475	63.66	75.14	65.40

Dual-camera MDP

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View1	57.98	63.09	53.64	78.68	92.54	1.43	68	31	36	0	623	2093	172	257	70.58	74.83	72.31
View2	53.12	52.43	53.84	80.20	78.09	4.33	66	44	22	0	1888	1662	155	206	55.85	75.54	57.67
Overall	55.63	57.67	53.73	79.38	85.20	2.88	134	75	58	0	2511	3755	327	463	63.79	75.16	65.58

Dual-camera MDP - Unbalanced Optimal Transport - model (a)

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View 1	53.68	59.51	48.89	76.27	92.85	1.32	68	30	38	0	577	2330	164	263	68.72	74.82	70.37
View 2	52.62	52.21	53.04	79.41	78.17	4.27	66	42	23	1	1861	1728	146	208	55.49	75.55	57.21
Overall	53.16	55.76	50.80	77.71	85.30	2.80	134	72	61	1	2438	4058	310	471	62.62	75.16	64.31

Dual-camera MDP - Unbalanced Optimal Transport - model (b)

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View1	55.49	60.41	51.31	78.42	92.32	1.47	68	31	37	0	640	2119	183	278	70.03	74.86	71.87
View2	51.89	51.37	52.42	79.83	78.23	4.28	66	41	23	2	1864	1693	146	207	55.88	75.50	57.59
Overall	53.75	55.83	51.82	79.07	85.18	2.87	134	72	60	2	2504	3812	329	485	63.51	75.16	65.30

Dual-camera MDP - Unbalanced Optimal Transport - pos

View	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
View 1	58.68	63.53	54.52	79.23	92.32	1.48	68	33	35	0	647	2039	167	269	70.94	74.66	72.62
View 2	55.82	55.29	56.36	79.49	77.98	4.32	66	40	23	3	1884	1721	145	226	55.32	75.35	57.02
Overall	57.30	59.38	55.37	79.35	85.09	2.90	134	73	58	3	2531	3760	312	495	63.74	74.98	65.44

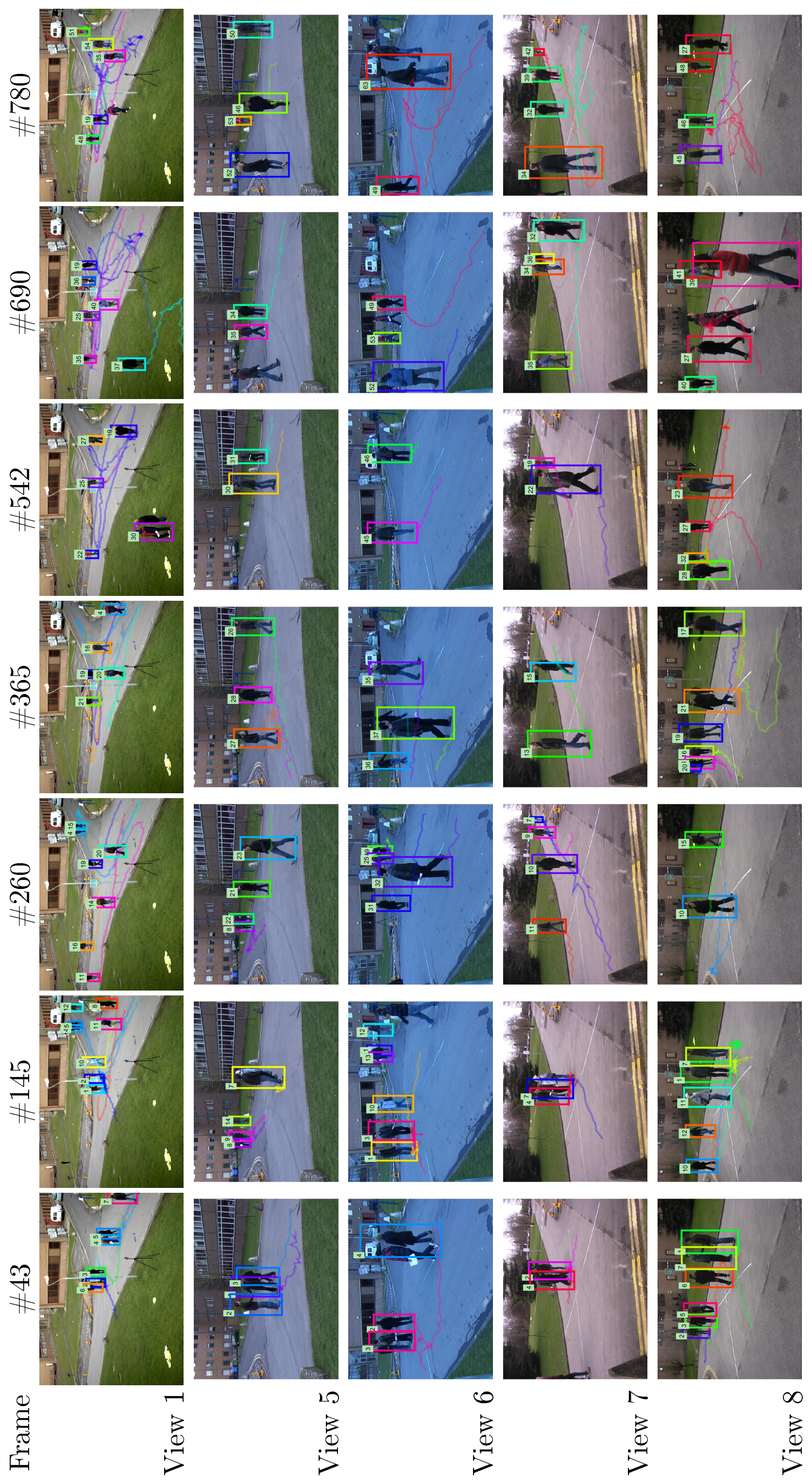


Figure A.1 – Visualization of the multi-camera collaborative tracking results of PETS09-S2L1 sequence

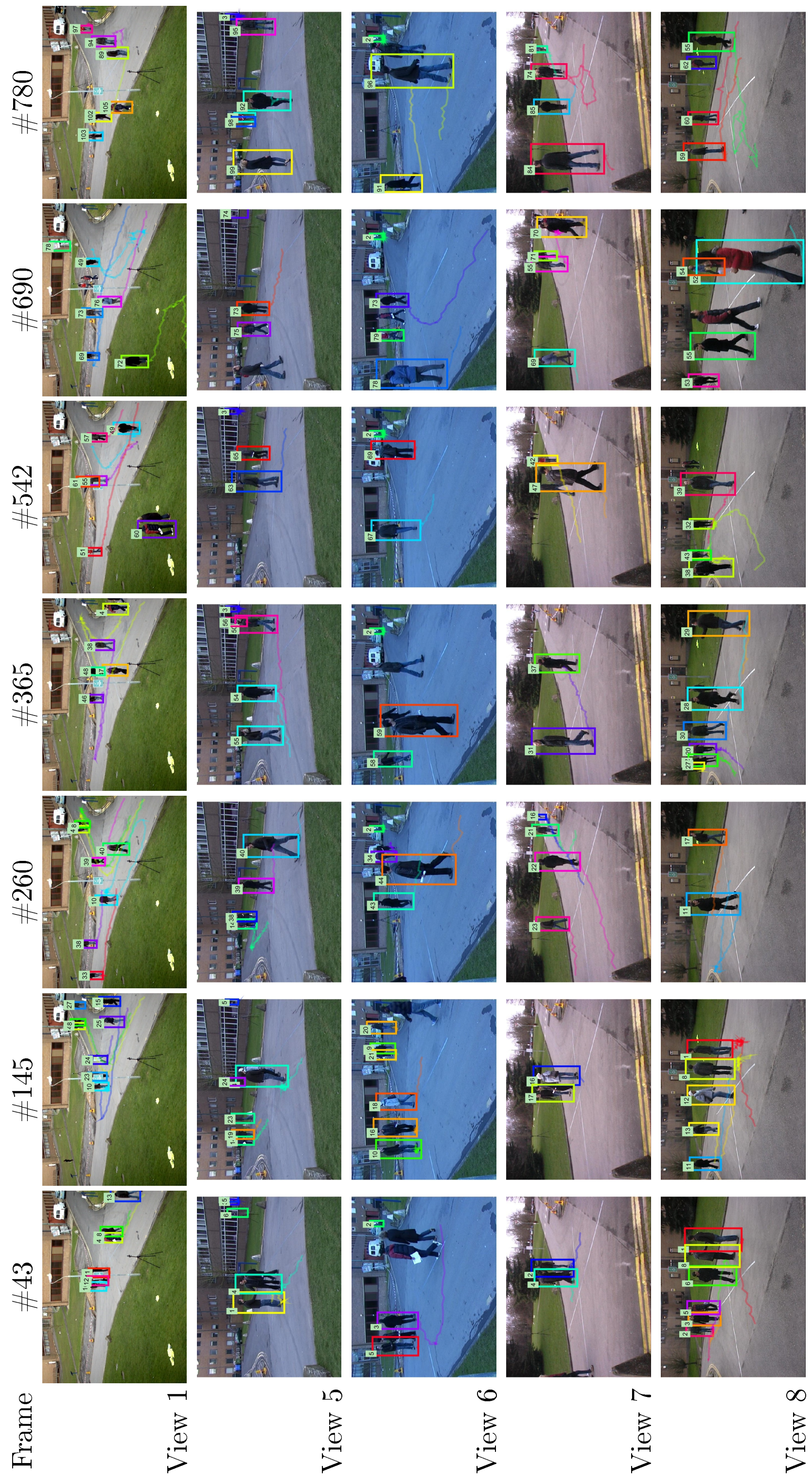


Figure A.2 – Visualization of the MDP tracking results of PETS09-S2L1 sequence

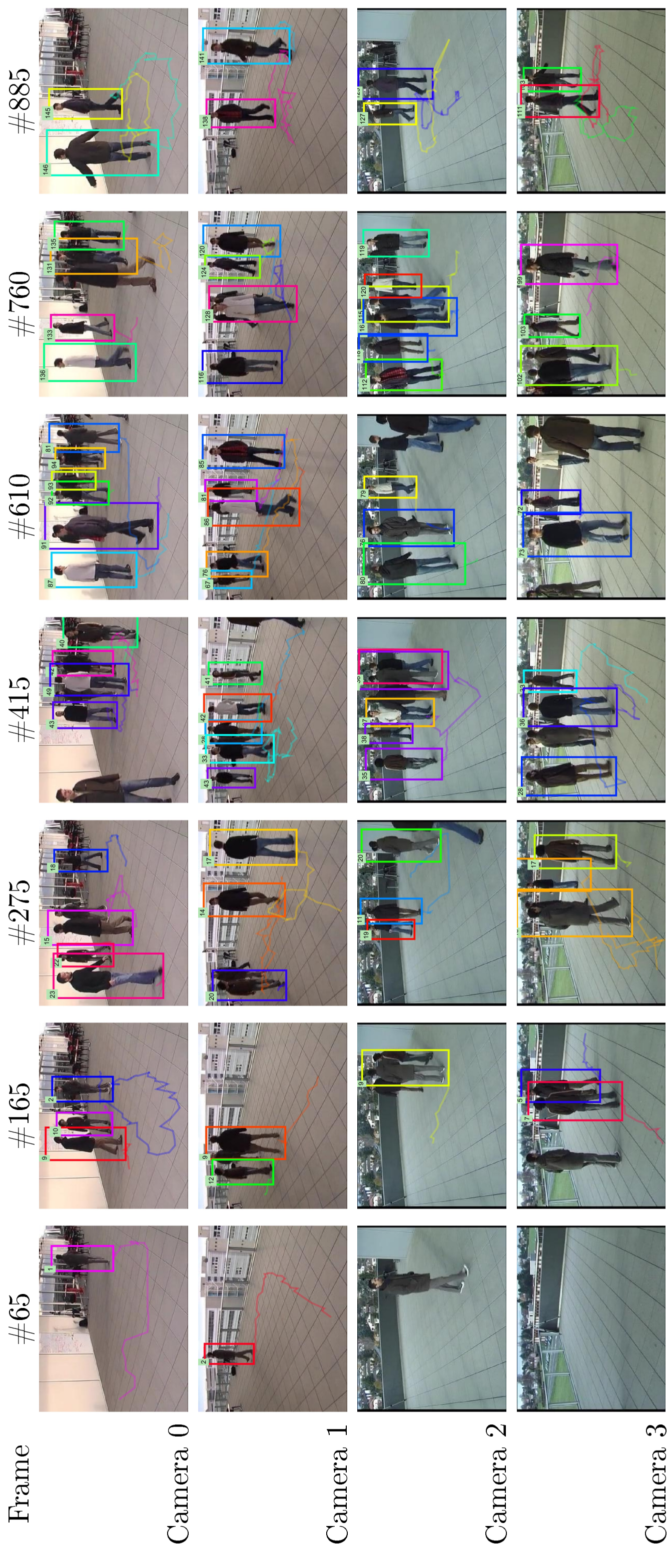


Figure A.3 – Visualization of the multi-camera collaborative tracking results of Terrace1 sequence

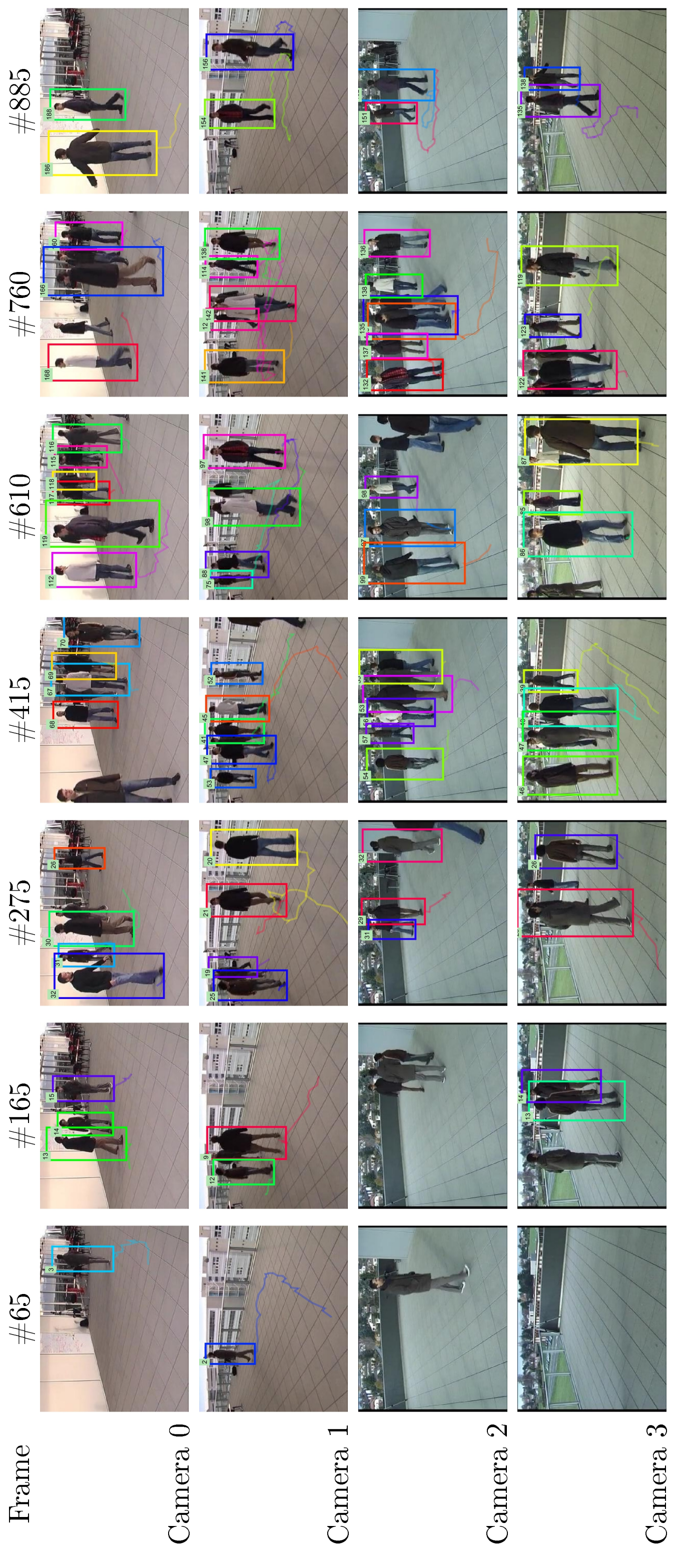


Figure A.4 – Visualization of the MDP tracking results of Terrace1 sequence

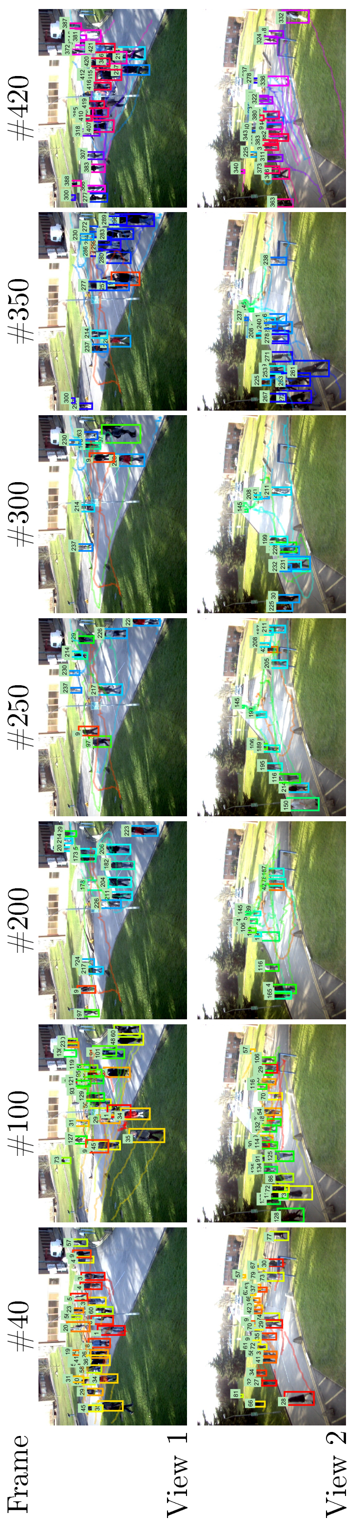


Figure A.5 – Visualization of the single-camera tracking results of PETS09-S2L2 sequence

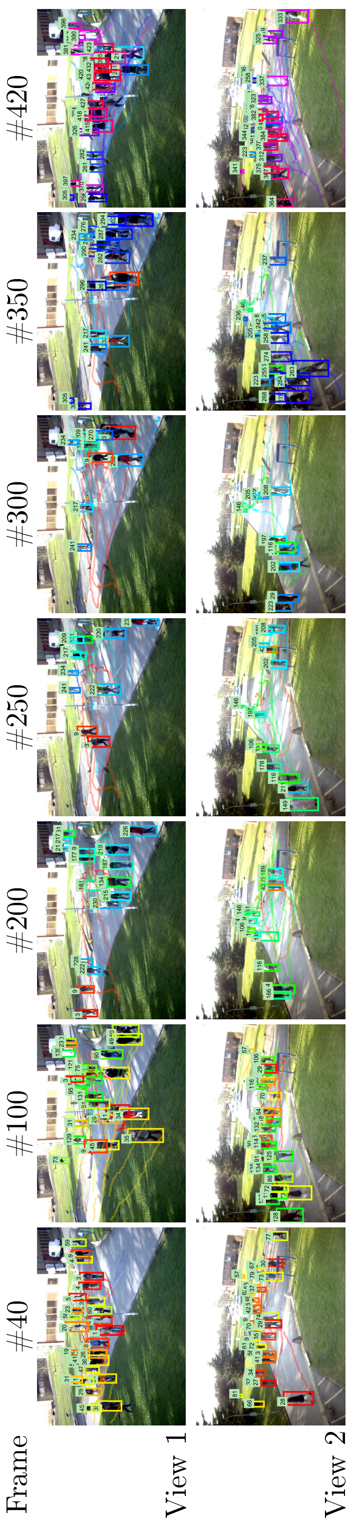


Figure A.6 – Visualization of the multi-camera collaborative tracking results of PETS09-S2L2 sequence

Bibliography

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[2] Jon Almazan, Bojana Gajic, Naila Murray, and Diane Larlus. Re-id done right: towards good practices for person re-identification. *arXiv preprint arXiv:1801.05339*, 2018.

[3] Mojtaba Amini-Omam, Farah Torkamani-Azar, and Seyed Ali Ghorashi. Maximum likelihood estimation for multiple camera target tracking on grassmann tangent subspace. *IEEE transactions on cybernetics*, 48(1):77–89, 2018.

[4] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. In *2008 IEEE Conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[5] Nadeem Anjum and Andrea Cavallaro. Trajectory association and fusion across partially overlapping cameras. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 201–206. IEEE, 2009.

[6] Shai Avidan. Ensemble tracking. *IEEE transactions on pattern analysis and machine intelligence*, 29(2):261–271, 2007.

[7] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1619–1632, 2011.

[8] Shai Bagon and Meirav Galun. Large scale correlation clustering optimization. *arXiv preprint arXiv:1112.2903*, 2011.

[9] Davide Baltieri, Roberto Vezzani, and Rita Cucchiara. Learning articulated body models for people re-identification. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 557–560. ACM, 2013.

[10] Davide Baltieri, Roberto Vezzani, and Rita Cucchiara. Mapping appearance descriptors on 3d body models for people re-identification. *International Journal of Computer Vision*, 111(3):345–364, 2015.

[11] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1-3):89–113, 2004.

[12] Chenglong Bao, Yi Wu, Haibin Ling, and Hui Ji. Real time robust l1 tracker using accelerated proximal gradient approach. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1830–1837. IEEE, 2012.

[13] Pierre Baqué, François Fleuret, and Pascal Fua. Deep occlusion reasoning for multi-camera multi-target detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 271–279, 2017.

[14] Yaakov Bar-Shalom, Thomas E Fortmann, Tracking, Data Association, et al. Vol. 179. *Mathematics in science and engineering*, 1988.

[15] Igor Barros Barbosa, Marco Cristani, Barbara Caputo, Aleksander Rognhaugen, and Theoharis Theoharis. Looking beyond appearances: Synthetic training data for deep cnns in re-identification. *Computer Vision and Image Understanding*, 167:50–62, 2018.

[16] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[17] Apurva Bedagkar-Gala and Shishir K Shah. Multiple person re-identification using part based spatio-temporal color appearance model. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1721–1728. IEEE, 2011.

[18] Apurva Bedagkar-Gala and Shishir K Shah. Part-based spatio-temporal model for multi-person re-identification. *Pattern Recognition Letters*, 33(14):1908–1915, 2012.

[19] Apurva Bedagkar-Gala and Shishir K Shah. A survey of approaches and trends in person re-identification. *Image and Vision Computing*, 32(4):270–286, 2014.

[20] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1806–1819, 2011.

[21] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 941–951, 2019.

[22] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *Journal on Image and Video Processing*, 2008:1, 2008.

[23] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[24] James Black, Tim Ellis, and Paul Rosin. Multi view image surveillance and tracking. In *Workshop on Motion and Video Computing, 2002. Proceedings.*, pages 169–174. IEEE, 2002.

[25] Michael J Black and Allan D Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[26] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550. IEEE, 2010.

[27] Jean-Yves Bouguet et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.

[28] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[29] William Brendel, Mohamed Amer, and Sinisa Todorovic. Multiobject tracking as maximum weight independent set. In *CVPR 2011*, pages 1273–1280. IEEE, 2011.

[30] Ted J Broida and Rama Chellappa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):90–99, 1986.

[31] Robert Grover Brown, Patrick YC Hwang, et al. *Introduction to random signals and applied Kalman filtering*, volume 3. Wiley New York, 1992.

[32] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European conference on computer vision*, pages 282–295. Springer, 2010.

[33] Quin Cai and Jake K Aggarwal. Tracking human motion in structured environments using a distributed-camera system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1241–1247, 1999.

[34] Yinghao Cai and Gerard Medioni. Exploring context information for inter-camera multiple target tracking. In *IEEE Winter Conference on Applications of Computer Vision*, pages 761–768. IEEE, 2014.

[35] Lijun Cao, Weihua Chen, Xiaotang Chen, Shuai Zheng, and Kaiqi Huang. An equalised global graphical model-based approach for multi-camera object tracking. *arXiv preprint arXiv:1502.03532*, 2015.

[36] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[37] Visesh Chari, Simon Lacoste-Julien, Ivan Laptev, and Josef Sivic. On pairwise costs for network flow multi-object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5537–5545, 2015.

[38] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[39] Kuan-Wen Chen, Chih-Chuan Lai, Pei-Jyun Lee, Chu-Song Chen, and Yi-Ping Hung. Adaptive learning for target tracking and true linking discovering across multiple non-overlapping cameras. *IEEE Transactions on Multimedia*, 13(4):625–638, 2011.

[40] Xiaojing Chen, Le An, and Bir Bhanu. Multitarget tracking in nonoverlapping cameras using a reference set. *IEEE Sensors Journal*, 15(5):2692–2704, 2015.

[41] Xiaotang Chen, Kaiqi Huang, and Tieniu Tan. Direction-based stochastic matching for pedestrian recognition in non-overlapping cameras. In *2011 18th IEEE International Conference on Image Processing*, pages 2065–2068. IEEE, 2011.

[42] Dong Seon Cheng and Marco Cristani. Person re-identification by articulated appearance matching. In *Person Re-Identification*, pages 139–160. Springer, 2014.

[43] Dong Seon Cheng, Marco Cristani, Michele Stoppa, Loris Bazzani, and Vittorio Murino. Custom pictorial structures for re-identification. In *Bmvc*, volume 1, page 6. Citeseer, 2011.

[44] Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced optimal transport problems. *Mathematics of Computation*, 87(314):2563–2609, 2018.

[45] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3029–3037, 2015.

[46] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005.

[47] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *2017 IEEE International Conference on Computer Vision (ICCV).(Oct 2017)*, pages 4846–4855, 2017.

[48] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):564–575, 2003.

[49] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[50] Serhan Coşar and Müjdat Çetin. A sparsity-driven approach to multi-camera tracking in visual sensor networks. In *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 407–413. IEEE, 2013.

[51] Council of European Union. Council regulation (EU) no 269/2014, 2014. `http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1416170084502&uri=CELEX:32014R0269`.

[52] Ingemar J. Cox and Sunita L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 18(2):138–150, 1996.

[53] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.

[54] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[55] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[56] Shahar Daliyot and Nathan S Netanyahu. A framework for inter-camera association of multi-target trajectories by invariant target models. In *Asian Conference on Computer Vision*, pages 372–386. Springer, 2012.

[57] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. *arXiv preprint arXiv:1811.07628*, 2018.

[58] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: efficient convolution operators for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6638–6646, 2017.

[59] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Convolutional features for correlation filter based visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 58–66, 2015.

[60] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 4310–4318, 2015.

[61] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision*, pages 472–488. Springer, 2016.

[62] Abir Das, Anirban Chakraborty, and Amit K Roy-Chowdhury. Consistent re-identification in a camera network. In *European conference on computer vision*, pages 330–345. Springer, 2014.

[63] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[64] Afshin Dehghan, Shayan Modiri Assari, and Mubarak Shah. Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4091–4099, 2015.

[65] Carlos R del Blanco, Raúl Mohedano, Narciso Garcia, Luis Salgado, and Fernando Jaureguizar. Color-based 3d particle filtering for robust tracking in heterogeneous environments. In *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–10. IEEE, 2008.

[66] Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

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

[68] Thanh-Toan Do, Toan Tran, Ian Reid, Vijay Kumar, Tuan Hoang, and Gustavo Carneiro. A theoretically sound upper bound on the triplet loss for improving the efficiency of deep distance metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10404–10413, 2019.

[69] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, 2014.

[70] A. Doucet, N. De Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[71] Wei Du and Justus Piater. Multi-camera people tracking by collaborative particle filters and principal axis-based integration. In *Asian Conference on Computer Vision*, pages 365–374. Springer, 2007.

[72] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.

[73] Ran Eshel and Yael Moses. Homography based multiple camera detection and tracking of people in a dense crowd. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

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

[75] Loïc Fagot-Bouquet, Romaric Audigier, Yoann Dhome, and Frédéric Lerasle. Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In *European Conference on Computer Vision*, pages 774–790. Springer, 2016.

[76] Michela Farenzena, Loris Bazzani, Alessandro Perina, Vittorio Murino, and Marco Cristani. Person re-identification by symmetry-driven accumulation of local features. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2360–2367. IEEE, 2010.

[77] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[78] J Ferryman and A Shahrokni. Pets2009: Dataset and challenge. In *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*, pages 1–6. IEEE, 2009.

[79] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-Ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. *arXiv preprint arXiv:1810.08278*, 2018.

[80] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690, 2019.

[81] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):267–282, 2008.

[82] Thomas Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE journal of Oceanic Engineering*, 8(3):173–184, 1983.

[83] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[84] Kyle A Gallivan, Anuj Srivastava, Xiuwen Liu, and Paul Van Dooren. Efficient algorithms for inferences on grassmann manifolds. In *IEEE Workshop on Statistical Signal Processing, 2003*, pages 315–318. IEEE.

[85] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. *arXiv preprint arXiv:1706.00292*, 2017.

[86] Andrew Gilbert and Richard Bowden. Tracking objects across cameras by incrementally learning inter-camera colour calibration and patterns of activity. In *European conference on computer vision*, pages 125–136. Springer, 2006.

[87] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[88] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[89] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *2011 international conference on computer vision*, pages 999–1006. IEEE, 2011.

[90] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *Bmvc*, volume 1, page 6, 2006.

[91] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *European conference on computer vision*, pages 234–247. Springer, 2008.

[92] Douglas Gray and Hai Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *European conference on computer vision*, pages 262–275. Springer, 2008.

[93] Seyed Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint probabilistic data association revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 3047–3055, 2015.

[94] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015.

[95] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2096–2109, 2016.

[96] Ben Harwood, BG Kumar, Gustavo Carneiro, Ian Reid, Tom Drummond, et al. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2829, 2017.

[97] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[98] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[99] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[100] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[101] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European conference on computer vision*, pages 702–715. Springer, 2012.

[102] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[103] R Henschel, L Leal-Taix, D Cremers, and B Rosenhahn. A novel multi-detector fusion framework for multi-object tracking. *arXiv preprint arXiv:1705.08314*, pages 1–10, 2017.

[104] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[105] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *CoRR*, abs/1703.07737, 2017.

[106] Martin Hirzer, Peter M Roth, Martin Köstinger, and Horst Bischof. Relaxed pairwise learned metric for person re-identification. In *European conference on computer vision*, pages 780–793. Springer, 2012.

[107] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[108] Carine Hue, J-P Le Cadre, and Patrick Pérez. Tracking multiple objects with particle filtering. *IEEE transactions on aerospace and electronic systems*, 38(3):791–812, 2002.

[109] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *European conference on computer vision*, pages 343–356. Springer, 1996.

[110] Omar Javed, Khurram Shafique, Zeeshan Rasheed, and Mubarak Shah. Modeling inter-camera space–time and appearance relationships for tracking across non-overlapping views. *Computer Vision and Image Understanding*, 109(2):146–162, 2008.

[111] Allan D Jepson, David J Fleet, and Thomas F El-Maraghi. Robust online appearance models for visual tracking. *IEEE transactions on pattern analysis and machine intelligence*, 25(10):1296–1311, 2003.

[112] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang. Visual tracking via adaptive structural local sparse appearance model. In *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*, pages 1822–1829. IEEE, 2012.

[113] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018.

[114] Hao Jiang, Sidney Fels, and James J Little. A linear programming approach for multiple object tracking. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[115] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. Pn learning: Bootstrapping binary classifiers by structural constraints. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 49–56. IEEE, 2010.

[116] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012.

[117] Gabin Kayumbi and Andrea Cavallaro. Multiview trajectory mapping using homography with lens distortion correction. *EURASIP Journal on Image and Video Processing*, 2008(1):145715, 2008.

[118] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970.

[119] Margret Keuper, Siyu Tang, Yu Zhongjie, Bjoern Andres, Thomas Brox, and Bernt Schiele. A multi-cut formulation for joint segmentation and tracking of multiple objects. *arXiv preprint arXiv:1607.06317*, 2016.

[120] Saad M Khan and Mubarak Shah. Tracking multiple occluding people by localizing on multiple scene planes. *IEEE transactions on pattern analysis and machine intelligence*, 31(3):505–519, 2009.

[121] Sohaib Khan and Mubarak Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1355–1360, 2003.

[122] Zia Khan, Tucker Balch, and Frank Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE transactions on pattern analysis and machine intelligence*, 27(11):1805–1819, 2005.

[123] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4696–4704, 2015.

[124] Kyungnam Kim and Larry S Davis. Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering. In *European Conference on Computer Vision*, pages 98–109. Springer, 2006.

[125] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[126] Martin Koestinger, Martin Hirzer, Paul Wohlhart, Peter M Roth, and Horst Bischof. Large scale metric learning from equivalence constraints. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2288–2295. IEEE, 2012.

[127] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[128] Harold W Kuhn. Variants of the hungarian method for assignment problems. *Naval Research Logistics Quarterly*, 3(4):253–258, 1956.

[129] Cheng-Hao Kuo, Chang Huang, and Ram Nevatia. Inter-camera association of multi-target tracks by on-line learned appearance affinity models. In *European Conference on Computer Vision*, pages 383–396. Springer, 2010.

[130] Junseok Kwon and Kyoung Mu Lee. Visual tracking decomposition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1269–1276. IEEE, 2010.

[131] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[132] Quoc Cuong Le, Donatello Conte, and Moncef Hidane. Online multiple view tracking: Targets association across cameras. In *6th Workshop on Activity Monitoring by Multiple Distributed Sensing (AMMDS 2018)*, 2018.

[133] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942 [cs]*, April 2015. arXiv: 1504.01942.

[134] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015.

[135] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[136] Karel Lenc and Andrea Vedaldi. R-cnn minus r. *arXiv preprint arXiv:1506.06981*, 2015.

[137] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

[138] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M Rehg. Video segmentation by tracking many figure-ground segments. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2192–2199, 2013.

[139] Peixia Li, Boyu Chen, Wanli Ouyang, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Gradnet: Gradient-guided network for visual object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6162–6171, 2019.

[140] Wei Li, Xiatian Zhu, and Shaogang Gong. Person re-identification by deep joint learning of multi-loss classification. *arXiv preprint arXiv:1705.04724*, 2017.

[141] Yuan Li, Chang Huang, and Ram Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2953–2960. IEEE, 2009.

[142] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-head r-cnn: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*, 2017.

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

[144] Baiyang Liu, Lin Yang, Junzhou Huang, Peter Meer, Leiguang Gong, and Casimir Kulikowski. Robust and fast collaborative tracking with two stage sparse optimization. *Computer vision–ECCV 2010*, pages 624–637, 2010.

[145] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*, 2018.

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

[147] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[148] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[149] David G Lowe et al. Object recognition from local scale-invariant features. In *iccv*, volume 99, pages 1150–1157, 1999.

[150] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[151] Stéphane Mallat. *A wavelet tour of signal processing.* Elsevier, 1999.

[152] Niki Martinel, Christian Micheloni, and Gian Luca Foresti. Saliency weighted features for person re-identification. In *European Conference on Computer Vision*, pages 191–208. Springer, 2014.

[153] Lain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):810–815, 2004.

[154] Xue Mei and Haibin Ling. Robust visual tracking using l(1) minimization. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1436–1443. IEEE, 2009.

[155] Xue Mei and Haibin Ling. Robust visual tracking and vehicle classification via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(11):2259–2272, 2011.

[156] Xue Mei, Haibin Ling, Yi Wu, Erik Blasch, and Li Bai. Minimum error bounded efficient l1 tracker with occlusion detection. In *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*, pages 1257–1264. IEEE, 2011.

[157] Alexis Mignon and Frédéric Jurie. Pcca: A new approach for distance learning from sparse pairwise constraints. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2666–2672. IEEE, 2012.

[158] Anton Milan, Laura Leal-Taixé, Konrad Schindler, and Ian Reid. Joint tracking and segmentation of multiple targets. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 5397–5406. IEEE, 2015.

[159] Anurag Mittal and Larry S Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo. In *European conference on computer vision*, pages 18–33. Springer, 2002.

[160] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017.

[161] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[162] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2016.

[163] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain monte carlo data association for multi-target tracking. *IEEE Transactions on Automatic Control*, 54(3):481–497, 2009.

[164] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):971–987, 2002.

[165] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[166] Nam Trung Pham, Weimin Huang, and SH Ong. Probability hypothesis density approach for multi-camera multi-object tracking. In *Asian Conference on Computer Vision*, pages 875–884. Springer, 2007.

[167] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR 2011*, pages 1201–1208. IEEE, 2011.

[168] Horst Possegger, Sabine Sternig, Thomas Mauthner, Peter M Roth, and Horst Bischof. Robust real-time tracking of multiple objects by volumetric mass densities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2395–2402, 2013.

[169] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6450–6458, 2019.

[170] Wei Qu, Dan Schonfeld, and Magdi Mohamed. Distributed bayesian multiple-target tracking in crowded environments using multiple collaborative cameras. *EURASIP Journal on Applied Signal Processing*, 2007(1):21–21, 2007.

[171] Richard J Radke. A survey of distributed computer vision algorithms. In *Handbook of Ambient Intelligence and Smart Environments*, pages 35–55. Springer, 2010.

[172] Christopher Rasmussen and Gregory D Hager. Joint probabilistic techniques for tracking multi-part objects. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*, pages 16–21. IEEE, 1998.

[173] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[174] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[175] Donald Reid. An algorithm for tracking multiple targets. *IEEE transactions on Automatic Control*, 24(6):843–854, 1979.

[176] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[177] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Deep-matching: Hierarchical deformable dense matching. *International Journal of Computer Vision*, 120(3):300–323, 2016.

[178] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, pages 17–35. Springer, 2016.

[179] Ergys Ristani and Carlo Tomasi. Tracking multiple people online and in real time. In *Asian Conference on Computer Vision*, pages 444–459. Springer, 2014.

[180] Ergys Ristani and Carlo Tomasi. Features for multi-target multi-camera tracking and re-identification. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[181] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 77(1-3):125–141, 2008.

[182] Sylvain Rousseau, Pierre Chainais, and Christelle Garnier. Dictionary learning for a sparse appearance model in visual tracking. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 4506–4510. IEEE, 2015.

[183] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[184] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *arXiv preprint arXiv:1701.01909*, 4(5):6, 2017.

[185] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[186] Vali Salari and Ishwar K. Sethi. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):87–91, 1990.

[187] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[188] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[189] Rajiv Shah and Jeremy Braithwaite. Spread too thin: analyzing the effectiveness of the chicago camera network on crime. *Police practice and research*, 14(5):415–427, 2013.

[190] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, 1993.

[191] Guang Shu, Afshin Dehghan, Omar Oreifej, Emily Hand, and Mubarak Shah. Part-based multiple-person tracking with partial occlusion handling. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1815–1821. IEEE, 2012.

[192] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 118–126, 2015.

[193] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[194] Paolo Spagnolo, P Mazzeo, and Cosimo Distante. *Human Behavior Understanding in Networked Sensing*. Springer, 2014.

[195] Santhoshkumar Sunderrajan and Bangalore S Manjunath. Multiple view discriminative appearance modeling with imcmc for distributed tracking. In *Distributed Smart Cameras (ICDSC), 2013 Seventh International Conference on*, pages 1–7. IEEE, 2013.

[196] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[197] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561, 2013.

[198] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[199] Murtaza Taj and Andrea Cavallaro. Multi-view multi-object detection and tracking. In *Computer Vision*, pages 263–280. Springer, 2010.

[200] Siyu Tang, Bjoern Andres, Miykhaylo Andriluka, and Bernt Schiele. Subgraph decomposition for multi-target tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5033–5041, 2015.

[201] Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Multi-person tracking by multicut and deep matching. In *European Conference on Computer Vision*, pages 100–111. Springer, 2016.

[202] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3539–3548, 2017.

[203] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders. Siamese instance search for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1420–1429, 2016.

[204] Yonatan Tariku Tesfaye, Eyasu Zemene, Andrea Prati, Marcello Pelillo, and Mubarak Shah. Multi-target tracking in multiple non-overlapping cameras using constrained dominant sets. *arXiv preprint arXiv:1706.06196*, 2017.

[205] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[206] Roger Y Tsai. An efficient and accurate camera calibration technique for 3d machine vision. *Proc. of Comp. Vis. Patt. Recog.*, pages 364–374, 1986.

[207] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[208] Jack Valmadre, Luca Bertinetto, João Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2805–2813, 2017.

[209] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[210] Cor J Veenman, Marcel JT Reinders, and Eric Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):54–72, 2001.

[211] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1:511–518, 2001.

[212] Paul Viola, Michael J Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[213] B-N Vo and W-K Ma. The gaussian mixture probability hypothesis density filter. *IEEE Transactions on signal processing*, 54(11):4091–4104, 2006.

[214] Jiuqing Wan and Liu Li. Distributed optimization for global data association in non-overlapping camera networks. In *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–7. IEEE, 2013.

[215] Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Online object tracking with sparse prototypes. *IEEE transactions on image processing*, 22(1):314–325, 2013.

[216] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017.

[217] Naiyan Wang, Jingdong Wang, and Dit-Yan Yeung. Online robust non-negative dictionary learning for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 657–664, 2013.

[218] Ning Wang, Yibing Song, Chao Ma, Wengang Zhou, Wei Liu, and Houqiang Li. Unsupervised deep tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1308–1317, 2019.

[219] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[220] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[221] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013.

[222] Brandon C Welsh and David P Farrington. *Crime prevention effects of closed circuit television: a systematic review*, volume 252. Citeseer, 2002.

[223] Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Citeseer, 2009.

[224] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[225] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

[226] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):210–227, 2009.

[227] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017.

[228] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.

[229] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *2015 IEEE international conference on computer vision (ICCV)*, number EPFL-CONF-230283, pages 4705–4713. IEEE, 2015.

[230] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 521–528, 2003.

[231] Jiarui Xu, Yue Cao, Zheng Zhang, and Han Hu. Spatial-temporal relation networks for multi-object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3988–3998, 2019.

[232] Yihong Xu, Yutong Ban, Xavier Alameda-Pineda, and Radu Horaud. Deepmot: A differentiable framework for training multiple object trackers. *arXiv preprint arXiv:1906.06618*, 2019.

[233] Fan Yang, Zhuolin Jiang, and Larry S Davis. Online discriminative dictionary learning for visual tracking. In *IEEE Winter Conference on Applications of Computer Vision*, pages 854–861. IEEE, 2014.

[234] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[235] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2015.

[236] Amir Roshan Zamir, Afshin Dehghan, and Mubarak Shah. Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. In *Computer Vision–ECCV 2012*, pages 343–356. Springer, 2012.

[237] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[238] Shengping Zhang, Hongxun Yao, Xin Sun, and Xiusheng Lu. Sparse coding based visual tracking: Review and experimental comparison. *Pattern Recognition*, 46(7):1772–1788, 2013.

[239] Shu Zhang, Elliot Staudt, Tim Faltemier, and Amit K Roy-Chowdhury. A camera network tracking (camnet) dataset and performance baseline. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 365–372. IEEE, 2015.

[240] Shu Zhang, Yingying Zhu, and Amit Roy-Chowdhury. Tracking multiple interacting targets in a camera network. *Computer Vision and Image Understanding*, 134:64–73, 2015.

[241] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Robust visual tracking via multi-task sparse learning. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2042–2049. IEEE, 2012.

[242] Tianzhu Zhang, Si Liu, Narendra Ahuja, Ming-Hsuan Yang, and Bernard Ghanem. Robust visual tracking via consistent low-rank sparse learning. *International Journal of Computer Vision*, 111(2):171–190, 2015.

[243] Tianzhu Zhang, Si Liu, Changsheng Xu, Shuicheng Yan, Bernard Ghanem, Narendra Ahuja, and Ming-Hsuan Yang. Structural sparse tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 150–158, 2015.

[244] Xuan Zhang, Hao Luo, Xing Fan, Weilai Xiang, Yixiao Sun, Qiqi Xiao, Wei Jiang, Chi Zhang, and Jian Sun. Alignedreid: Surpassing human-level performance in person re-identification. *arXiv preprint arXiv:1711.08184*, 2017.

[245] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

[246] Rui Zhao, Wanli Ouyang, and Xiaogang Wang. Unsupervised salience learning for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3586–3593, 2013.

[247] Liang Zheng, Yi Yang, and Alexander G Hauptmann. Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984*, 2016.

[248] Zhedong Zheng, Liang Zheng, and Yi Yang. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3754–3762, 2017.

[249] Zhang Zhipeng, Peng Houwen, and Wang Qiang. Deeper and wider siamese networks for real-time visual tracking. *arXiv preprint arXiv:1901.01660*, 2019.

[250] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. Robust object tracking via sparsity-based collaborative model. In *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*, pages 1838–1845. IEEE, 2012.

[251] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*, 2017.

[252] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 366–382, 2018.

Résumé :

Le travail présenté dans cette thèse abordent les problématiques de détection et suivi d'objets, en utilisant un système de caméras collaboratives. L'idée principale de l'utilisation de plusieurs caméras pour réaliser le suivi est de résoudre les problèmes d'occultation que les méthodes de suivi de mono-caméra sont incapables de régler. Pour s'adapter des critères dans plusieurs applications de surveillance, nos travaux se concentrent sur le problème de suivi en ligne de plusieurs objets dans le contexte de plusieurs caméras synchronisées et dont les champs de vue sont chevauchent. Dans le cas de notre étude, les axes suivants ont été étudiés : premièrement, utiliser plusieurs caméras pour suivre une seule cible; deuxièmement, suivre plusieurs d'objets simultanément; finalement, réidentifier les objets qui réapparaissent ultérieurement dans le champs de vue. Dans les conditions où les tâches de suivi se font sur une scène en plein air, l'apparence des objets change à cause des conditions de luminosité variant à l'extérieur, ou des mouvements des objets eux-mêmes. Souvent, les performances de suivi sont dégradées par les algorithmes défaillants dus à la perte de leurs cibles. Nous avons développé des algorithmes de suivi avec multi-caméras qui permettent à chaque caméra de participer au processus de suivi des autres caméras dans le réseau.

Notre algorithme a été évalué par les métriques communnes sur les bases de données publiques. Les résultats expérimentaux ont montré la pertinence de nos algorithmes de suivi multi-caméras par rapport à une seule caméra, ainsi que l'impact des différentes caractéristiques sur la performance de suivi de notre approche de suivi par multi-caméras.

Mots clés :

Détection d'objets, Suivi de mono-objet, Suivi de multi-objet, Suivi de multi-target multi-camera, Apprentissage profond de caractéristiques, Transport optimal.

Abstract :

The work presented in this thesis concerns the problem of visual multiple object tracking using a system of collaborative cameras. The main idea of using a multi-camera system in tracking is to solve occlusion problems, which single-camera tracking methods are unable to solve. With multiple automated surveillance applications in mind, our work focuses on the problem of online multi-object tracking in a multi-camera system in which fields of view are overlapped, and video frames are synchronized. In the case of our study, the thesis includes the following objectives: firstly, to track a single object in a multi-camera system; secondly, to track multiple objects simultaneously; finally, to re-identify objects which possibly reenter the fields of view of the cameras multiple times. In outdoor tracking scenes, objects often change their appearance, including their shape, their size, and their texture, due to the varying lighting condition and the movement of the objects themselves. This causes tracking algorithms to lose their targets frequently, and therefore degrade tracking performance. We developed multi-camera tracking algorithms that allow each camera to participate in the overall tracking process of the network to improve its tracking results.

Our algorithm by the common metrics on public multi-camera video databases. Our experiments showed the relevance of our multi-camera tracking algorithms to single-camera ones, as well as the impacts of different characteristic features on the tracking performance of our multi-camera tracking approach.

Keywords :

Object Detection, Single-Object Tracking, Multi-Object Tracking, Multi-Target Multi-Camera Tracking, Deep Feature Learning, Optimal Transport.