People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
8 May 1945 Guelma University

Faculty of Mathematics, Computer Science and Material Sciences
Department of Computer Science
Laboratory of Information and Communication Sciences and Technologies

DISSERTATION

In view of obtaining
The Doctorate degree in 3rd cycle

Domain: Mathematics and Computer Science. **Field**: Computer Science
Specialty: Computer Science

Presented by:

BENRAZEK Ala Eddine

Entitled

Internet of Things: Analysis of suspicious behaviour in a surveillance camera network

Defended: ……. Before the board of examiners composed of:

Pr. Yacine LAFIFI	**Professor**	Univ. of 8 May 1945, Guelma	Chairman
Dr. Brahim FAROU	**MCA**	Univ. of 8 May 1945, Guelma	Supervisor
Pr. Hamid SERIDI	**Professor**	Univ. of 8 May 1945, Guelma	Co-supervisor
Pr. Mohamed BENMOHAMMED	**Professor**	Univ. of Abdelhamid Mehri, Constantine	Examiner
Pr. Nadir FARAH	**Professor**	Univ. of Badji Mokhtar, Annaba	Examiner
Pr. Mohamed Tarek KHADIR	**Professor**	Univ. of Badji Mokhtar, Annaba	Examiner
Dr. Zineddine KOUAHLA	**MCA**	Univ. of 8 May 1945, Guelma	Invited

2020-2021

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God for giving me the strength, knowledge, ability, and opportunity to undertake this research study and persevere and complete it satisfactorily.

I would like to sincerely thank my thesis director, Dr. Brahim FAROU, for my graduate study's outstanding orientation. Thank you for teaching and advising me for the four years. Your vital advice helped me map out my following years as a postgraduate student, building up my future research career. Thank you for trusting me and for everything you have done to bring me to this point.

I owe a lot to my co-director Professor Hamid SERIDI who has helped, encouraged and guided me a lot for several years. I thank him for his numerous remarks and appreciations of extreme relevance, and I would like to express my deep gratitude to him on this occasion.

I wish to express my deep gratitude to Professor Yacine LAFIFI of the University of Guelma, to Professor Mohamed BENMOHAMMED of the University of Constantine and to Professors Nadir FARAH and Tarek KHADIR of the University of Annaba for the honour they have bestowed on me by accepting the responsibility of examining this work and participating in the defence jury.

I owe a lot to Dr. Zineddine KOUAHLA, who, especially in the last two years, has given me the benefit of his immense experience and encouraged me to progress in my work. I thank him infinitely for his help, and I express my deepest gratitude.

I would like to express my deep gratitude to Dr. Mohamed Amine FERRAG for his invaluable help and support at the technical level, especially during this past year.

I would also like to thank all the Computer Science department teachers of 8 Mai 1945 Guelma university for the quality of their formation. A big thanks to the LabSTIC laboratory teams who contributed, each in their way, and I do not forget the laboratory engineer Miss Madiha KHAROUBI.

My sincere thanks to my friends and colleagues, especially Rochdi BOUDJAHEM and Gattal ELHACHEMI.

ملخص

يركز العمل المقدم في أطروحة الدكتوراه هذه على تطوير نظام مراقبة بالفيديو موزع على نطاق واسع لتتبع الأجسام المتحركة المشبوهة وتحليل سلوكها في بيئة إنترنت الأشياء. أنظمة المراقبة بالفيديو الى يومنا هذا لا تزال تواجه مشكلة زيادة البيانات غير الضرورية والمتكررة الناجمة عن تكرار كشف نفس الأحداث بواسطة عدة كاميرات بسبب تداخل مجال الرؤية. لا تؤثر هذه المشكلات على الزيادة في موارد المعالجة والتخزين والاتصال المستهلكة فحسب ، بل تؤثر أيضا على جودة التتبع وجودة التحليل السلوكي للأشياء المتعقبة وتشغيل النظام في الوقت الفعلي. تتفاقم هذه التأثيرات بشكل خاص بسبب عمليات النشر الكثيفة لشبكات الكاميرات واسعة النطاق. لمعالجة هذه المشكلة، اقترحنا نظاما جديدا موزعا وتعاونيا لتتبع الأجسام المتحركة. يهدف هذا النظام إلى تحسين جودة المراقبة وتقليل تكلفة معالجة البيانات من خلال تقليل عدد الكاميرات النشطة. يعمل النظام المقترح على مرحلتين: ١) انتخاب قائد من بين مجموعة من الكاميرات الدي يملك أفضل رأية. ٢) اختيار أفضل المساعدين من الكاميرات المجاورة لتعظيم الاكتشاف عندما تكون رؤية القائد غير كافية لتتبع الكائن. فقط القائد ومساعدوه ينشطون في عملي التتبع. الكاميرات المجاورة الأخرى تظل في حالة غير نشطة. لتحسين معالجة البيانات في الوقت الفعلي، تم توزيع حمل النظام عبر بنية حوسبة إنترنت فيديو الأشياء(IoVT) . اقترحنا أيضا طريقتين لتجميع الكاميرات بناء على معيار مساحة تداخل مجال الرؤية بدلا من معيار الراديو والمسافة للحد من تعقيد آلية التنسيق، وضمان جدوى تشغيلها في الشبكات واسعة النطاق، وتقييد نطاق التواصل بين الكاميرات. تعتمد التقنية المقترحة الأولى على خوارزمية التصنيف الهرمي التصاعدي (AHC) . تركز هذه الطريقة بشكل أساسي على تجميع الكاميرات ذات الحد الأقصى من التداخل. لسوء الحظ، تمتلك الطريقة معرفة جزئية عن الشبكة وحالتها، أي أنها تعرف فقط الحد الأقصى من التداخل بين زوجين من الكاميرات ولا تؤخذ التداخلات الأخرى في الاعتبار ويتم إهمالها تماما. لهذا الغرض، اقترحنا تقنية تجميع ثانية تسمح لنا بتجميع ليس فقط الكاميرتين اللتين تتداخلان بشكل كبير ولكن أيضا جميع الكاميرات التي تتداخل مع أكبر عدد ممكن من الكاميرات. للعثور على هذه المجموعة، استخدمنا خوارزمية البحث Bron-karboch. يتيح لنا هذا النهج العثور على الحد الأقصى من Clique، أي الحد الأقصى لعدد الكاميرات المتداخلة مع بعضها البعض. في الأخير، اقترحنا آلية فهرسة جديدة وفعالة تعتمد على هيكل الشجرة. تهدف الآلية المقترحة إلى فهرسة البيانات الضخمة الناتجة عن شبكة الكاميرات واسعة النطاق لتنظيمها بشكل مناسب لتقليل وقت البحث قدر الإمكان لضمان تشغيل النظام في الوقت الفعلي. تعتمد هذه البنية على التقسيم المتكرر للبيانات باستخدام خوارزمية التجميع k-means لفصل البيانات بشكل فعال إلى مجموعات فرعية غير متداخلة لتحسين نتائج خوارزميات البحث والاكتشاف. تظهر النتائج التي تم الحصول عليها فعالية الطرق المقترحة من حيث جودة المراقبة وكمية البيانات على الشبكة واستهلاك الطاقة والتشغيل في الوقت الحقيقي مقارنة بالنظام التقليدي.

الكلمات المفتاحية إنترنت فيديو الأشياء ، الحوسبة السحابية ، نظام المراقبة بالفيديو الموزعة ، تتبع الكائنات ، الذكاء الاصطناعي ، إدارة البيانات والفهرسة.

RÉSUMÉ

Le travail présenté dans cette thèse de doctorat vise à développer un système de vidéosurveillance distribué à grande échelle pour suivre les objets suspects et analyser leur comportement dans un environnement IoT. À ce jour, les systèmes de vidéosurveillance restent confrontés au problème des données inutiles et redondantes causées par la détection multiple des mêmes événements par plusieurs caméras en raison du chevauchement des champs de vision. Ces problèmes affectent non seulement l'augmentation des ressources de traitement, de stockage et de communication consommées, mais aussi la qualité du suivi, la qualité de l'analyse comportementale des objets suivis et le fonctionnement en temps réel du système. Ces effets sont particulièrement exacerbés par les déploiements denses de réseaux de caméras à grande échelle. Pour résoudre ces problèmes, nous avons proposé un nouveau système de suivi distribué et collaboratif. Ce système vise à améliorer la qualité de la surveillance et à réduire le coût du traitement des données en réduisant le nombre de caméras actives et en réduisant les données communicantes. Le système proposé fonctionne en deux étapes : (*i*) Élection d'un chef qui a la meilleure vision de l'objet détecté parmi les caméras voisines. (*ii*) Sélection des meilleurs assistants des caméras voisines pour maximiser la détection lorsque la vision du chef est insuffisante pour suivre l'objet. Seul le chef et ses assistants sont actifs. Les autres caméras voisines restent inactives. Pour améliorer le traitement des données en temps réel, la charge du système est répartie sur l'architecture de l'IoVT. Nous avons proposé deux méthodes de regroupement des caméras basées sur le critère de la surface de chevauchement du champ de vision plutôt que sur le critère de la radio et de la distance afin de réduire la complexité du mécanisme de coordination, d'assurer la faisabilité de leur fonctionnement dans des réseaux à grande échelle et de limiter la plage de communication entre les caméras. La première technique proposée est basée sur l'algorithme de classification hiérarchique ascendante. Cette méthode se concentre principalement de regrouper les caméras qui ont un chevauchement maximal. Malheureusement, la méthode ne dispose que de connaissances partielles sur le réseau et son état, c'est-à-dire qu'elle ne connaît que le chevauchement maximum. Les autres chevauchements ne sont pas pris en compte et sont complètement négligés. Pour surmonter cette limite, nous avons proposé une deuxième technique de regroupement qui

permet de regrouper non seulement les deux caméras qui se chevauchent le plus, mais aussi toutes les caméras qui se chevauchent avec le plus grand nombre de caméras possible. Pour trouver ce groupe, nous avons utilisé l'algorithme de recherche basé sur la Clique de Bron-karboch. Nous avons également proposé un nouveau mécanisme d'indexation efficace basé sur la structure arborescente. Le mécanisme proposé vise à indexer les données massives générées par le réseau de caméras à grande échelle afin de les organiser intelligemment pour réduire au maximum le temps de recherche et assurer le fonctionnement du système en temps réel. Cette structure est basée sur le partitionnement récursif de données à l'aide de l'algorithme de regroupement k-means pour séparer efficacement les données en sous-groupe bien séparé afin d'améliorer les résultats des algorithmes de recherche et de découverte. Les résultats obtenus démontrent l'efficacité des méthodes proposées en termes de qualité du suivi, de quantité de données sur le réseau, de consommation d'énergie et de fonctionnement en temps réel par rapport au système conventionnel.

Mots-clés : Internet des Objets Vidéo, Cloud-Fog Computing, Système de Vidéo Surveillance Distribuée, Suivi d'Objets, Intelligence Artificielle, Gestion et Indexation de Données.

ABSTRACT

The work presented in this Ph.D. thesis focuses on developing a large-scale distributed video surveillance system for tracking suspicious moving objects and analysing their behaviour in an IoT environment. To date, video surveillance systems still face the problem of unnecessary and redundant data caused by multiple detections of the same events by several cameras due to overlapping fields of view. These problems affect not only the increase in processing, storage, and communication resources consumed but also the quality of the tracking, the quality of the behavioural analysis of the tracked objects, and the real-time operation of the system. These effects are particularly exacerbated by dense deployments of large-scale camera networks. To address these issues, we proposed a new distributed and collaborative tracking system. This system aims to improve the quality of monitoring and reduce the cost of data processing by reducing the number of active cameras and reducing communicating data. The proposed system operates in two steps: (*i*) Electing a leader who has the best view of the detected object among the neighbouring cameras. (*ii*) Choosing the best assistants from neighbouring cameras to maximise detection when the leader's vision is insufficient to track the object. Only the leader and their assistants are active. The other neighbouring cameras remain in an inactive state. To improve real-time data processing, the system's load is distributed throughout the IoVT computing architecture. We proposed two methods of grouping cameras based on the FoV overlap area criterion instead of the radio and distance criterion to reduce the coordination mechanism's complexity, ensure the feasibility of their operation in large-scale networks, and restrict the communication range between cameras. The first proposed technique is based on the ascending hierarchical classification algorithm. This method mainly focuses on grouping cameras that have maximum overlap. Unfortunately, the method has only partial knowledge about the network and its state, i.e., it only knows the maximum overlap. The other overlaps are not taken into account and are completely neglected. To exceed this limit, we proposed a second grouping technique that groups not only the two most overlapping cameras but also all overlapping cameras with as many cameras as possible. To find this group, we used the Bron-karboch Clique-based search algorithm. We also proposed a new and efficient indexing

mechanism based on the tree structure. The proposed mechanism aims to index the massive data generated by large-scale cameras network to organise it appropriately to reduce the search time as much as possible to ensure real-time system operation. This structure is based on recursive partitioning of space using the k-means clustering algorithm to effectively separate space into non-overlapping subspace to improve search and discovery algorithm results. The results obtained demonstrate the effectiveness of the proposed methods in terms of tracking quality, amount of network data, energy consumption and real-time operation compared to the conventional system.

Key-words: Internet of Video Things, Cloud-Fog Computing, Distributed Video Surveillance System, Object Tracking, Artificial Intelligence, Data Management and Indexing.

Contents

List of Figures

List of Tables

ABBREVIATIONS

AHC	Ascending Hierarchical Classification
BS	Base Station
CCN	Co-operative Camera Network
CCTV	Closed-Circuit TeleVision
CH	Cluster Head
CoT	Cloud of Things
DAI	Distributed Artificial Intelligence
DVR	Digital Video Recorder
DW	Data Warehouses
ECR	Energy Conservation Ratio
FIFO	First In First Out
Fog	From cOre to edGe
FoV	Field of View
IoE	Internet of Everything
IoST	Internet of Surveillance Things
IoT	Internet of Things
IoVT	Internet of Video Things
IoMT	Internet of Multimedia Things
IP	Internet Protocol
ITS	Intelligent Transportation Systems

ITU	International Telecommunication Union
IVSS	Intelligent Video Surveillance Systems
LAN	Local Area Network
MBR	Minimum Bounding Regions
MC3	Multi-Camera Coordination and Control
NIST	National Institute of Standards and Technology
NN	Nearest Neighbor
QoS	Quality of Service
RFID	Radio Frequency IDentification
UAV	Unmanned Aerial Vehicle
VANET	Vehicle Ad-hoc NETwork
WMSN	Wireless Multimedia Sensor Network
WSN	Wireless Sensor Network
WVSN	Wireless Visual Sensor Network
WWW	World Wide Web

INTRODUCTION

General context and issues

The closed-circuit television (CCTV) or video surveillance systems that have become part of the building infrastructure play a vital role in our lives due to their immense benefits in improving our community's safety. The task of monitoring, tracking, and observing numerous moving targets distributed around a large-scale environment (e.g., airport terminals, railway and subway stations, bus depots, shopping malls, school campuses, military bases, etc.) is the main focus of the surveillance problem. Conventional surveillance systems consist of a large number of cameras placed in a large-scale surveillance area connected to a centralised based station. Unfortunately, manually controlling all cameras to find interest targets by qualified human operators is very difficult, especially as the number of cameras increases, making this mission almost impossible. To address these issues, two particular aspects of artificial intelligence have appeared. The first is the development of computer vision algorithms. This latter can process large amounts of visual data and offer human-like input on the scene under surveillance's evolution. Consequently, traditional surveillance systems have been replaced by modern, more innovative surveillance systems known as Intelligent Video Surveillance Systems (IVSS). Artificial intelligence, pattern recognition, and computer vision technology have given IVSS systems the ability to automatically recognise irregular behaviours and patterns in a video. The second aspect is advancing in distributed computing and distributed intelligence. These latter allow cameras to be treated as distinct entities, capable of adapting to the evolution of the scene and the complexity of the communication network, reducing the bandwidth of information and deriving a better interpretation of people's dynamics and objects moving in the scene. Sadly, the cameras are autonomous; operate without any knowledge of these neighbours' distribution or functioning because of the absence of communication and coordination between them. This absence leads to loss of useful information and redundant event detection, which means that more redundant multimedia data is generated and, therefore, more computing power, more storage space, and more bandwidth to process, store and transmit it.

In the last few years, new strategies have focused on the Internet of Things (IoT) and their modern paradigms (cloud computing, fog computing, mist computing, etc.). This new concept, which designates the interconnection of a set of objects via an Internet network, has made it possible to retrieve information of very great importance at the semantic level, which was a task

for human operators until now. Together with a modern computing paradigm of IoT, smart cameras formed the next generation of video surveillance systems called the "Internet of Video Thing or IoVT". IoVT is a part of the IoT capable of efficiently processing large volumes of data, such as images and videos. The interconnection of surveillance cameras allows a constant and unlimited exchange of relevant information on the nature of objects and their behaviour in an environment that is no longer limited by the field of vision of a camera. However, this large amount of information managed by the camera network has made the old techniques quickly obsolete in the face of new challenges. A new mechanism capable of managing a set of cameras coupled with the IoVT concept to detect, recognise, track, and interpret objects' behaviour is necessary to overcome these emerging problems. Therefore, efficient collaboration and data management mechanism is needed to control and coordinate these cameras in real-time, which is the main focus of this thesis.

Objectives

This thesis will design a novel and efficient, intelligent video surveillance system that effectively leverages the IoVT infrastructure to track objects in real-time. The long-term objective of this system is to address the following central problem:

> ***"How to coordinate a multi-smart camera network to track a set of moving targets with minimal cost, high tracking quality, and in real-time?"***

Tracking a set of targets in real-time, with minimal cost and high quality, is indispensable in large-scale video surveillance. Coordinating multi-cameras to tracking these targets with these constraints is challenging and non-trivial. This is mainly due to the following practical questions:

- *The trade-off between maximising the quality of target tracking and minimising the number of active cameras*: The increase in the number of active cameras for target tracking may increase the quality of this mission's results. Still, it is necessary to consider the redundancy of the captured data, which negatively influences real-time operation and increases the system's necessary resources. Reducing active cameras can reduce the data generated, therefore reducing resources and decreasing the quality of the tracking targets. Consequently, it is necessary to address this trade-off.
- *Scalable communication*: Collaborating between cameras to track targets is an effective way to improve system efficiency. Despite their benefits, the latter's complexity makes developing a coordination mechanism intricate, especially in large-scale multi-camera networks. Also, excessive communication is needed, which increases significantly as the number of cameras increases, necessitating a wide bandwidth and increased energy consumption. Therefore, the coordination framework needs to be scalable depending on the communication overload between the cameras.
- *Flexible and robust infrastructure*: The development of a decentralised video surveillance system is required after the conventional central system's failure. For this purpose, we need a flexible and robust infrastructure to support a large-scale distributed video surveillance system.
- *Real-time performance*: These smart cameras' control decisions should be taken efficiently, effectively, and in real-time. The management of smart cameras' information to detect and track moving objects is necessary to accomplish these objectives.

As a result, the smart camera network's coordination and collaboration to track targets is an intricate problem requiring many studies.

Scientific Contributions

In this PhD thesis, several contributions are made to achieve the desired system. The major contribution can be summarised as follows:

- ✍ **First contribution**: Our review of the literature found that video surveillance systems use several IoVT computing paradigms. These various paradigms are used based on the needs of the system's tasks. Consequently, we have proposed a new classification for video surveillance systems according to the paradigm(s) used.
- ✍ **Second contribution**: Our literature review on data indexing techniques has led us to propose a new taxonomy of existing indexing mechanisms and their related structures, through which we qualitatively compare the associated works.
- ✍ **Third contribution**: We have introduced a new distributed architecture based on the IoVT computing paradigm to improve real-time object tracking quality in a video surveillance system. The proposed architecture consists of five layers: Visual sensors layer, Mist layer, Fog layer, Cloud layer, and Application layer. Each layer has specific tasks among the tasks of real-time tracking of moving objects. These tasks are assigned according to the task requirements and the resources available in the layer nodes. For example, time-sensitive data is executed at the Mist layer, or in the worst case, at the Fog layer if there is insufficient Mist capacity to perform the task. On the other hand, data analysis is assigned at the Cloud level because it requires many computing and storage resources.
- ✍ **Fourth contribution**: We have proposed two mechanisms for grouping cameras based on the FoV overlap surface criterion instead of the radio or distance criterion between cameras. The first approach uses the ascending hierarchical classification algorithm, while the second uses the Bron-karboch Clique-based search algorithm. Our approaches aim to create highly overlapping groups of cameras to restrict communication and coordination only at the group level and not for all system cameras. The idea is that if the area of FoV overlap between the cameras is relatively large, the cameras will act similarly in terms of coverage. Therefore, we will group them into a single group. Our approaches can significantly simplify the coordination process, help to avoid redundant event detection between overlapping cameras, and ensure system operation in large-scale networks. Consequently, they play a vital role in conserving energy and enhancing the network's lifetime by enabling collaboration capabilities between the group's cameras and avoiding redundant detection and processing.
- ✍ **Fifth contribution**: We have proposed a new distributed and collaborative tracking system based on the modern computing paradigm of IoVT. The system's primary purpose is to increase network life, reduce processing costs, decrease communication data, and improve tracking quality. The proposed system operates in two steps: (*i*) Electing a leader among a set of cameras grouped according to the overlap degree of their FoVs. (*ii*) Choosing the best assistants from neighbouring cameras maximises detection when the leader's vision is insufficient to track the object. Only the leader and their assistants are active. The other neighbouring cameras remain in an inactive state. To improve

real-time data processing, the system's load is distributed throughout the modern IoVT infrastructure.

✍ **Sixth contribution**: We have proposed a new data management mechanism based on the indexing technique for our video surveillance system in particular and IoT data in general. This proposal aims to improve objects' search and discovery strategy quality to re-identify them during the tracking. The proposed index structure called *B*inary tree based on *C*ontainers at *C*loud-*F*og computing level (BCCF-tree). BCCF-tree structure benefits the emerging IoVT computing paradigms, representing the most powerful real-time processing capacity provided by Fog computing due to its proximity to sensors and the largest storage capacity provided by Cloud computing.

Thesis Roadmap

This thesis is structured in two main parts. The first part is divided into three state-of-the-art chapters, while the second part is composed of two contribution chapters. The thesis concludes with a general conclusion.

❖ Part I: Backgrounds, Preliminaries and Basic Concepts

❑ *Chapter 01: "Evolution of Modern Computing Paradigms in the Internet of Things"*

This chapter reviews the technologies, relevant concepts, and definitions of the various modern computing paradigms emerging in the IoT. First, we review the Internet of Things (IoT) technology, its operating principles, and some current and future applications. Next, we introduce Cloud and Fog computing, as well as the related terminology. These paradigms are studied and analysed in depth. We identify their functional principles and highlight points of similarities and differences.

❑ *Chapter 02: "Intelligent Video Surveillance Systems -IVSS-"*

This chapter covers the current state of video surveillance and its development. It presents the most essential video surveillance system evolution. Following that, it presents the different components of a video surveillance system and the various image processing techniques used. It provides various classifications and potential issues of the video surveillance system. Next, it explains why video surveillance systems should use collaborative intelligence and a shared distributed framework. The chapter ends with a presentation of a peek into the future of intelligent video surveillance systems.

❑ *Chapter 03: "Indexing Data Techniques"*

This chapter presents the data produced by the IoT and video surveillance systems working as a main data source. First, it presents the data management process's overall cycle. Then, it concentrates on the management and indexing of massive data in the field of IoT. The required elements for understanding data indexing and searching in metric spaces as a data abstraction space are presented. In the end, it

presents a comprehensive review of current indexing strategies in the literature.

❖ **Part II: Proposed Collaborative Video Surveillance System for Suspicious Behaviour Analysis**

❑ *Chapter 04: "Distributed Collaborative Multi-Cameras for Tracking Moving Objects based on Internet of Video Things -IoVT- Paradigms"*

This chapter will describe the proposed architecture of a video surveillance system based on the IoVT paradigm. Then, it presents our proposed multi-camera grouping techniques and their results. Ultimately, it presents the proposed distributed and collaborative tracking system.

❑ *Chapter 05 "An Efficient Indexing for Massive Video Surveillance Data based on Internet of Video Things -IoVT- Paradigms"*

This chapter presents our architecture for the presented IoVT data indexing system. After that, it explains in detail the proposed index structure. Then show the evaluation of the proposed structure on several real datasets with the results.

Part I

Backgrounds, Preliminaries and Basic Concepts

CHAPTER 1

EVOLUTION OF MODERN COMPUTING PARADIGMS IN THE INTERNET OF THINGS

Chapter contents

1.1 Introduction

Today, it is very difficult to neglect the impact of technology on our world. After the World Wide Web or the Web in the 1990s, we are moving towards the potentially most "confusing" phase of the Internet revolution, namely the "Internet of Things" or "IoT". The IoT establishes a link between real and virtual objects, allowing connectivity anytime, anywhere, for anything and not just anyone. This has created a new world in which physical objects and beings and data and virtual environments interact with each other in the same place and at the same time [363]. Connected objects or devices have existed in one form or another since introducing the first computer networks. However, the idea of a globally connected planet began to take shape only when the Internet emerged.

The rapid development of IoT technologies has led to the emergence of stringent requirements, including data storage, data analysis, data exchange between devices, security and privacy, and unified and ubiquitous access. As a solution to these requirements, "Cloud computing" has emerged as a cost-effective alternative to owning reliable computing resources without owning any infrastructure. Cloud computing is a metaphor for the Internet [163]. It is a parallel and distributed system consisting of interconnected computers (physical and virtual) as a unified computing resource that provides data transmission, storage, and computing services to consumers on-demand and pay-per-use business model to reduce costs and improve profits.

Buyya and Dastjerdi [79], in their book "*Internet of Things*", estimate that around one trillion devices (such as mobile phones, tablets, sensors, motors, relays, and actuators) will be connected to the network by 2025. Moreover, the amount of data generated by these devices is expected to reach about 163 Zettabytes [118], leading to a "data tsunami" [169]. Transferring all this data for processing in the cloud will consume many bandwidth resources, and storage [295] will have a very high operating cost and increase the overall delay of data processing [338]. For this reason, "Fog computing" is introduced to improve load balancing in the cloud by allowing processing to be done locally [111]. Fog computing focuses on bringing intelligence, processing, and data storage closer to the edge of the network to resolve the limits of cloud computing by reducing the Internet latency, conserving the network bandwidth, enhancing the operational efficiency, speeding up the real-time processing, and storing of sensitive data close to where the data are generated [265].

This chapter reviews the technologies, relevant concepts, and definitions of the different modern computing paradigms emerging in IoT. First of all, in section 1.2, we will explain in detail the IoT technology, the principle of its use and especially the impact it will have on our way of life.

We will therefore present some existing and possible applications of IoT. Next, in section 1.3, we will present the cloud computing technology. In section 1.4, we will introduce the concept of fog computing, distinguish related terms and show the differences and similarities between them.

1.2 Internet of Things (IoT)

1.2.1 Origin and development of the concept of the IoT

After Tim Berners-Lee launched the WWW in 1991, a whole new world emerged thanks to Kevin Ashton and David L. Brock of the Auto-ID Lab at the Massachusetts Institute of Technology (MIT) coined the term Internet of Things in 1999. In [35], Ashton described his vision in which computers would collect data without human assistance and transform it into useful information. This would be made possible by sensors and Radio Frequency IDentification (RFID) that allow computers to observe, identify and understand the world.

> *"I could be wrong, but I'm fairly sure the phrase "Internet of Things" started life as the title of a presentation I made at Procter & Gamble (P&G) in 1999."*
> *"The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so."*
>
> *Kevin Ashton, 2009*

Later that year, Neil Gershenfeld published his book on Thinking Objects, where he sees WWW's evolution as a real explosion where objects start using the Internet so that people don't need them [153].

> *"in retrospect looks like the rapid growth of the World Wide Web may have been just the trigger charge that is not setting off the real explosion, as things start to use the Net."*
>
> *Neil Gershenfeld, 1999*

Simultaneously, the concept of "*Ubiquitous Computing*" was introduced by Mark Weiser in [457]. He developed a fundamental vision of future technological ubiquity, in which increasing "availability" of processing power would be accompanied by decreasing "visibility". As he explained, "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it".

> *"Ubiquitous computing is roughly the opposite of virtual reality, where virtual reality puts people inside a computer- generated world, ubiquitous computing forces the computer to live out here in the world with people."*
>
> *Mark Weiser, 1998-1999*

In the late 1990s, the number of hosts exceeded 2 million, while the number of sites reached 4 million [351]. At that period, the Institute of Electrical and Electronics Engineers (IEEE) published the 802.11b (Wi-Fi) standard, with a transmission speed of 11 Mbits/s. This technology

was the starting point for Wireless Sensor Networks (WSN) due to the large number of small distributed devices that can detect, record, and communicate physical data wirelessly [178].

At the beginning of the XXIst century, the IoT reached another level when the International Telecommunication Union (ITU) published its first report on the IoT, which considers it a first contribution to the definition and understanding of the IoT. This report envisions devices and all types of objects becoming active users of the Internet on humans' behalf. The connection of inanimate objects and things[1] to communication networks, in addition to the deployment of higher-speed mobile networks offering permanent connectivity, would make it possible to realise the vision of a truly ubiquitous network, "anytime, anywhere, by anyone and anything" [195].

> *"A new dimension has been added to the world of information and communication technologies (ICTs): from anytime, any place connectivity for anyone, we will now have connectivity for anything. Connections will multiply and create an entirely new dynamic network of networks – an Internet of Things."*
>
> *UIT, 2005*

Three years later, for the first time, a group of 50 companies launched the Internet Protocol for Smart Objects (IPSO) Alliance to promote the use of the Internet Protocol (IP) in intelligent object networks and enable the IoT.

In 2008, the number of devices connected to the Internet surpassed the world's population for the first time. These two years (2008-2009), according to Dave Evans, Cisco Futuristic Leader and Chief Technologist for Cisco IBSG, are the birth years of the IoT, where objects are connected to the Internet as people [135].

> *"..., Cisco IBSG estimates IoT was "**born**" sometime between 2008 and 2009 ..."*
>
> *Dave Evans, 2011*

1.2.2 IoT definition

Since the birth of IoT, several international organisations and research centres have participated in creating common standards for IoT. Establishing a common definition of IoT was one of the first steps in their process. The IoT has many definitions over this long history, but there is no standard, unified definition so far. For this reason, we have cited some of the most frequently cited definitions in the literature.

The meaning of IoT and the consequences of its deployment in our environments, according to Kevin Ashton, are as follows:

"*If we had computers that knew everything there was to know about things—using data they gathered without any help from us—we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best. We need to empower computers with their own*

[1] In this thesis, the two terms, "objects" and "things", are used identically. Other terms frequently used by the research community are "intelligent objects", "devices", "nodes".

means of gathering information, so they can see, hear and smell the world for themselves." (Kevin Ashton, [35])

In 2015, Study Group 20 (SG-20) was created due to the 10-year experience after the first ITU report on IoT in 2005. SG-20 defines IoT as:

"*A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.*" (SG-20, [476])

The Coordination and Support Action for Global RFID-related Activities and Standardisation (CASAGRAS) defined IoT in its final report as:

"*A global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. This infrastructure includes existing and evolving Internet and network developments. It will offer specific object-identification, sensor and connection capability as the basis for the development of independent cooperative services and applications. These will be characterised by a high degree of autonomous data capture, event transfer, network connectivity and interoperability.*" (CASAGRAS, [83])

IEEE-IoT has published a document with an overview of IoT applications and a proposed definition. The document presents two definitions of IoT and are as follows:

"*An IoT is a network that connects uniquely identifiable 'Things' to the Internet. The 'Things' have sensing/actuation and potential programmability capabilities. Through the exploitation of unique identification and sensing, information about the 'Thing' can be collected and the state of the 'Thing' can be changed from anywhere, anytime, by anything.*" (IEEE-IoT, [190])

The following definition refers to large-scale scenarios:

"*Internet of Things envisions a self-configuring, adaptive, complex network that interconnects 'things' to the Internet through the use of standard communication protocols. The interconnected things have physical or virtual representation in the digital world, sensing/actuation capability, a programmability feature and are uniquely identifiable. The representation contains information including the thing's identity, status, location or any other business, social or privately relevant information. The things offer services, with or without human intervention, through the exploitation of unique identification, data capture and communication, and actuation capability. The service is exploited through the use of intelligent interfaces and is made available anywhere, anytime, and for anything taking security into consideration.*" (IEEE-IoT, [190])

There are other definitions proposed in the literature by other authors in various research papers concerning research centres and laboratories. A major contribution is that of Atzori et al., who defined IoT as:

"*The Internet of Things (IoT) is a novel paradigm that is rapidly gaining ground in the scenario of modern wireless telecommunications. The basic idea of this concept is the pervasive presence around us of a variety of things or objects – such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals.*" (Atzori et al., [40, 158])

Mercedes Bunz and Graham Meikle, in their book published in 2018, gives the following definition:

" *The Internet of things describes the many uses and processes that result from giving a network address to a thing and fitting it with sensors. These conjunctures of sensors, things and networks have become an increasingly important part of internet experiences. When we equip the things around us with sensors and connect them to networks, they gain new capabilities ... we mean a particular ability that things did not have before - such as seeing, speaking to, or tracking people.*" (Mercedes Bunz & Graham Meikle, [76])

Based on all the definitions we have cited, in the following definition and Figure 1.1, we summarise the IoT definition.

" *The IoT is a network of physical (hardware) and virtual (software) objects, which are uniquely and unambiguously identifiable, operating in an intelligent environment. These objects can interact and communicate anywhere, anytime, and with anything through standardized communication protocols using information and ICT technologies to exchange data and information stored and processed by appropriate big data techniques.*"

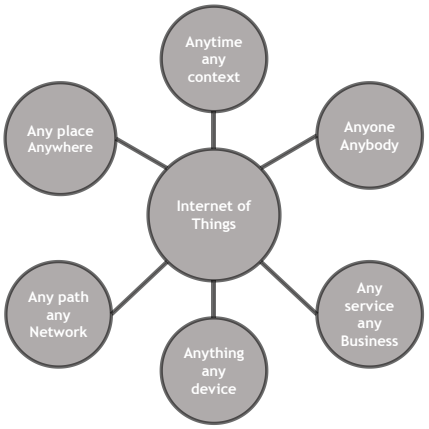


FIGURE 1.1: IoT definition (from [325])

1.2.3 From Internet of Things (IoT) to Internet of Everything (IoE)

According to Cisco [107], IoE is the result of increased network connectivity. IoT is only concerned with the interconnection of smart devices and objects in the aforementioned definitions, while IoE is more extensive. It focuses on the interconnection between people, data and (business) processes, and things to create new network connections and unleash unprecedented value [116]. This means that the IoT is a part or component of the IoE, as shown in Figure 1.2.

- **People**: according to Gartner [267], people themselves will become nodes on the Internet, connecting to the Internet using devices (such as sensors placed on the skin or sewn) and social networks [404].

> *"...people themselves will become nodes on the Internet, with both static information and aconstantly emitting activity system."*
>
> *Gartner, 2012*

- **Process**: processes based on artificial intelligence play an essential role in the cooperation of each of the entities of the IoE. The main objective of these processes is to extract the best out of the crucial data to transmit it to the right person, at the right time and in the right way [107].
- **Data**: the raw data generated by the connected devices are initially useless until they are analysed and processed in the data centre by artificial intelligence-based methods to extract more useful information for faster and more intelligent decisions, as well as to control our environment more efficiently [107, 356].

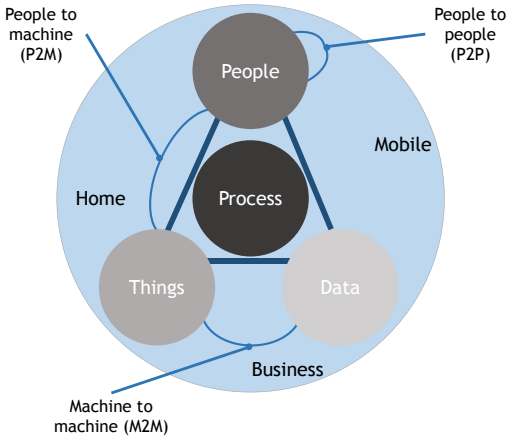


FIGURE 1.2: Internet of everything (IoE)

1.2.4 Architecture of IoT

Compared to the Internet that uses the TCP/IP protocol, the IoT connects billions of objects that will create much more traffic and require much more data storage in addition to security and governance [407]. Therefore, several factors, such as scalability, interoperability, reliability, quality of service (QoS), etc., must be considered in the proposed IoT architecture.

Like the unclear definition of the IoT, its architecture is also unclear. Several architectures have been proposed in the literature for independent applications of IoT. The two well-known architectures are (i) Three-layer architecture [493] and (ii) Five-layer architecture [407, 468]. In addition, there are other extensions of these architectures and which, for the sake of brevity, are presented in Table 1.1.

1.2.4.1 Three-layer architecture

The three-layer architecture was introduced in the early stages of research in this field and consisted of three layers: the perception layer, the network layer and the application layer. Each layer has a large scale of information with different enabling technologies and functionalities, as shown in Figure 1.3.

In summary, the Perception or Device layer is responsible for identifying each thing in the IoT system and collecting information about the environment. The Transport, Network, or

Transmission layer deals with network operations. This layer is responsible for connecting intelligent objects, network devices and servers and transmitting data collected by the previous layer to the next layer. The Application or Process layer uses intelligent computing technology such as data mining to extract significant information from massive data processing and provides an interface between the users and the IoT. There are various applications in which IoT can be deployed in this layer, including smart homes, smart cities, and smart health.

This hierarchical architecture's operating principle starts when the Perception layer collects data from connected objects using its detection or sensing technology, such as RFID tags, sensors, cameras, etc. It then transfers the gathered data to the Application layer, using Internet communication techniques through the Transport layer for processing and analysis, to store it in databases or share it with other application systems [471].

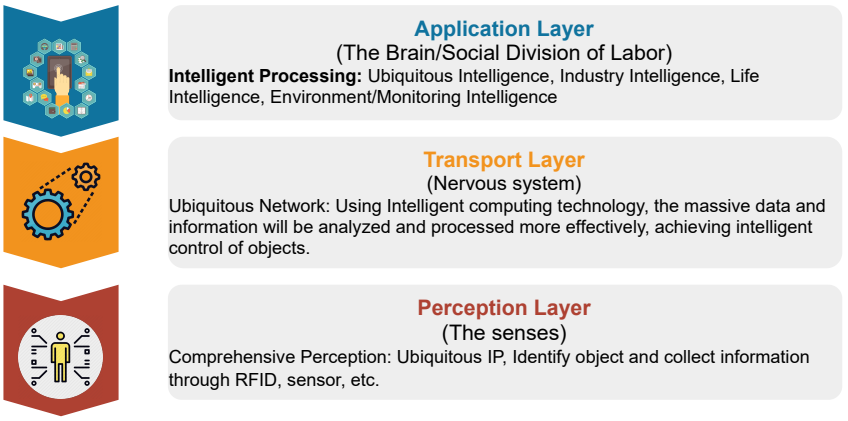


FIGURE 1.3: Three-layer architecture of the IoT

1.2.4.2 Five-layer architecture

With the rapid development of the IoT, the three-layer architecture has become insufficient to withstand this expansion. That is why a five-layer architecture has been proposed by adding two new layers of the processing and business layer, as shown in Figure 1.4. We outline the function of all layers in the following itemised list:

- ❏ *The perception layer*: the primary task of this layer is to collect information on the physical properties of objects such as temperature, air quality, speed, humidity, pressure, flow, movement and electricity, etc. using various devices such as smart cards, RFID tags, readers, and sensor networks and convert it into digital signals, which is more practical for its transmission over the network.
- ❏ *The network layer*: is the foundation layer of the IoT. It contains the Internet network's software and hardware equipment to connect the devices that enable the IoT environment. The main function of the transport layer is to transmit the data collected by the connected objects from the perception layer to the processing centre via various networks, such as wireless or wired networks, even the local area network (LAN). The main techniques of this layer are presented in Figure 1.4.
- ❏ *The middleware layer*: is also known as the processing layer. Each device implements its type of service and only connects and communicates with other devices that implement the same service. This layer consists of systems for storing, analysing and processing

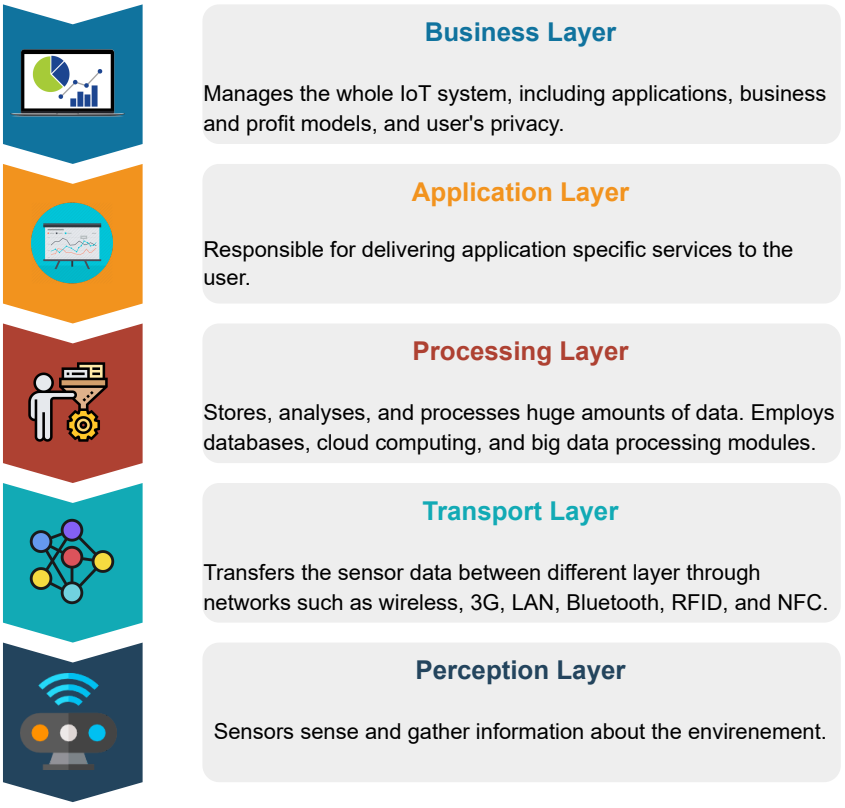


FIGURE 1.4: Five-layer architecture of the IoT

large amounts of data received from the transport layer to make an automatic decision. Moreover, for this purpose, this layer uses many technologies such as databases, big data processing techniques, ubiquitous computing, and cloud computing.

- ❏ *The application layer*: is responsible for offering application-specific services to the user based on the process layer's data. This layer plays an essential role in defining various applications in which the IoT can be deployed, such as smart health, smart agriculture, smart home, smart city, smart transportation, etc.
- ❏ *The business layer*: is the last layer in the IoT architecture. This layer's principal role is to define the load and control of IoT applications, including the management of applications, the relevant business model, and other activities such as user privacy and confidentiality.

For more details about IoT architectures, you can refer to [228, 362].

1.2.5 IoT applications

If we have seen all the efforts that have been made by researchers and industry to develop and improve the IoT from its birth to date from a general point of view, we can see that it has one objective, which is the building of a "*Smart City*". The smart city is one of the richest and most complex scenarios in IoT applications [159]. In general, it refers to a city that uses ICT technologies to improve the quality of urban services and reduce their costs. Its general purpose includes several domains, as shown in Figure 1.5.

TABLE 1.1: Extension of the IoT reference architecture

	Ref	Description — Object — Activities
Six layers architecture	[503]	The IoT's six-layer architecture adds a new layer at the bottom compared to the five-layer architecture called "Coding Layer". This layer ensures that a unique identification number is issued for each object to identify it throughout the IoT cycle.
	[412]	In addition to the five layers of the IoT reference architecture, a layer has been introduced under the name "MAC layers". MAC layer allows for monitoring and control of devices due to the resource limit of IoT devices. It makes the devices sleep during their idle period to save energy.
Seven layers architecture	[108]	According to the white paper of Cisco published in 2014, the seven-layer architecture allows the user to consider various factors that affect the sensing capabilities of sensors or actuators to consider the scalability and ensure supportability. Compared to the five-layer architecture, Cisco divides the "Middleware layer" into three layers, which are: • *Edge computing:* This layer focuses on convert network data flows, filtering, analysing and transforming high volumes of data. • *Data accumulation*: This layer concerns the storage and guarantee of data produced by different IoT devices. • *Data abstraction*: In this layer, the data is prepared for analysis to develop more straightforward and performance-enhancing applications.

FIGURE 1.5: Block diagram of a smart home system

1.2.5.1 Smart home

The smart home is the result of the cooperation of different connected sensors, trying to automate the daily tasks to improve the quality of life as much as possible. This purpose is achieved through intelligent services and decisions for resources management as well as security. Among the services developed for smart homes:

- *Detection systems*: permanent movement monitoring to know if there are individuals in the house to control the consumption of unnecessary resources such as lights, radiators, etc., and detect if a person has fallen like the elderly and disabled people thanks to pressure and movement sensors.
- *Safety monitoring*: video surveillance cameras can be used to record events of interest. These can then be used for feature extraction to find out what is happening in real-time. In addition to home alarm systems, people feel safe in their daily lives at home.
- *Remote-control systems*: the remote control of domestic machines such as washing machines, stoves, and refrigerators is a very practical and helpful way for people who work a lot, who have forgotten and physically handicapped people.

1.2.5.2 Smart environment/space and agriculture

A smart environment is the combination of a physical environment that consists of different autonomous or semi-autonomous technological devices and a data management infrastructure supported by ubiquitous computing that allows the analysis and processing of heterogeneous data collected continuously from the environment in real or quasi-real-time to provide intelligent services to support the daily activities of the community's users.

Below are some cases of IoT use in the field of agriculture and the environment:

- *Air pollution monitoring*: aims to establish a collaborative network between robots such as drones, pollution detection or measurement sensors equipped in it and ubiquitous computing for monitoring and controlling CO_2 emissions from factories, pollution from cars and toxic gases generated on farms.
- *Disaster detection*: with the difficulty of predicting natural disasters, intelligent IoT devices that act as a warning and monitoring system during natural disasters and operate through communication via the Internet can be crucial in avoiding many severe losses.
- *Smart farming*: in intelligent agriculture based on IoT, the crop field is monitored using sensors like light, humidity, temperature, soil moisture, etc., and irrigation system automation. Using this system, farmers can monitor field conditions from anywhere.
- *Livestock monitoring*: today, farmers can use wireless IoT technologies to collect their cattle's location and health data. This information helps them identify sick animals to prevent the spread of disease. It also reduces labour costs by enabling farmers to locate their cattle using IoT-based sensors.

1.2.5.3 Smart water management

The IoT does not stop at these points. With the collection of real-time information such as flow, velocity, temperature, water quality and water consumption by connecting the residential water meter and the floating sensor network [417] to the Internet, we can, among other things, reduce the cost of workforce and maintenance, improve accuracy and reduce water consumption, which is the main goal that a human being can achieve to conserve this essential part of life.

1.2.5.4 Smart transport

Smart transport systems are cooperative systems that enable vehicles to communicate with each other and their environment, such as the road, to improve management, operations, and user services through the combination of information and ubiquitous computing with transport networks.

- *Smart roads*: or smart highways are used to describe roads that use IoT technology and traffic analysis algorithms to make driving safer. Smart roads are divided into three main parts: road equipment that allows vehicle recognition through surveillance cameras, road maintenance, and connectivity and detection using smart sensors to detect everything like the weather and unexpected events such as accidents or traffic jams.
- *Smart cars*: are autonomous vehicles, independent of the driver's will, and require only the available external aids such as signs, traffic lights, road markings and information provided by smart roads to take control or decide on braking, acceleration, turning, etc.
- *Smart parking*: Based on the new IoT technologies, Smart parking provides an intelligent solution to quickly find a parking space in large cities to reduces parking search time, traffic, and fuel burn rate in addition to road safety. This system consists of installing intelligent sensors and micro-controllers to detect vehicles' presence or absence in each parking space. Thanks to a simple mobile/web application, this system can inform users in real-time about the availability or occupancy of spaces and reserve their places to ensure that other drivers don't take them.

1.2.5.5 Smart building

Smart buildings are traditional buildings that include a layer of IoT devices such as intelligent sensors, actuators and meters to retrieve a wide range of data such as temperature, humidity, sound level, presence or absence of people in a given space. With the technological innovations used by smart buildings, end-users benefit from improved comfort, safety and low-cost services [332].

1.2.5.6 Smart energy and smart grid

Smart Grid is one of the areas in which industry, governments, and academia are interested and investing significantly [145, 272]. IoT provides more information on electricity suppliers and consumers' behaviours in an automated way to improve energy efficiency. It also provides consumers with intelligent energy management, such as smart meters, smart appliances and renewable energy resources [37].

1.2.5.7 Smart healthcare

The deployment of IoT technologies and ubiquitous computing in the healthcare sector offers more opportunities to develop intelligent healthcare systems to improve real-time safety, simplify maintenance, reduce costs, and give patients more control over their lives and treatment at all times. Typically, systems collect vital patient data via a network of sensors connected to medical devices and ensure ubiquitous access or sharing of medical data.

1.3 Cloud computing (CC)

1.3.1 A brief history: from mainframes to clouds

This section will highlight some of the most significant technological development milestones that the world has gone through, from simple terminals/machines to robust and efficient networks and clouds (see Figure 1.6).

> *"Computing may someday be organized as a public utility just as the telephone system is a public utility."*
>
> *McCarthy in 1961*

FIGURE 1.6: Birth of cloud computing

Cloud computing is a typical development of a computing resource's time-sharing model between the 1950s and 1970s. Throughout this period, the companies began using dumb terminals to access powerful-shared *Mainframes* Mainframes that simultaneously provided services (storage, computing, etc.) to multiple users. These mainframes are not available to the public. Only big companies can use them because of their very high cost. In the 1980s, *Personal Computers* or *PCs* appeared at a lower cost. As a result, companies started using ordinary computers instead of mainframe computers, where it was not necessary to share the mainframe with someone else.

With the development of the Internet from a local to a *Global Network*, the increase in network access speed, and the development of IT technology, consumers have begun to access applications and resources remotely through the Internet using mobile devices. Thanks to this new connectivity, *Grid computing* was introduced in the second half of the 1990s as a new form of distributed computing. This paradigm's basic idea is to take advantage of unexploited computing power to increase computing power when needed.

> *"The Grid is an emerging infrastructure that will fundamentally change the way we think about -and use- computing"*
>
> *Ian Foster and Carl Kesselman in 1999*

The self-configuration, self-optimisation, self-healing and self-protection of autonomous systems has been one of the most important developments that have contributed to the emergence of cloud computing as an autonomous control of computer networks, witnessing the exploitation of the resources available on the Internet, on-demand, in a simple and scalable way. The first scientific user of the term cloud computing was Professor Ramnath Chellappa of the University of Texas at Austin in 1997. Salesforces, in 1999, was the first to transform this concept into business by starting to deliver enterprise applications via an easy-to-use website for enterprises as a service.

Between 1999 and 2009, the concept of cloud computing began to spread in many companies such as Amazon (Amazon Web Services in 2002 and Elastic Compute Cloud in 2006), Microsoft (Azure in 2010), IBM (IBM SoftLayer and IBM Bluemix in 2005) or Google (Google Cloud Platform in 2008).

Based on this brief background, cloud computing is not a new technology. As shown in Figure 1.7, cloud computing is a major evolution of four technology areas and is not, as some believe, a simple hardware development [78].

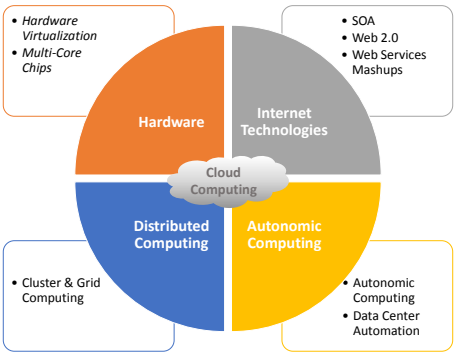


FIGURE 1.7: Various advances leading to the emergence of cloud computing

1.3.2 Cloud computing definition

The ambiguity surrounding the concept of cloud computing makes it logical to find several definitions for this concept. At present, many researchers in industry and academia have tried to define "cloud computing".

The first academic definition of cloud computing was coined by Pr. Ramnath Chilapa of Emory University and the University of Southern California in a talk titled Intermediaries in Cloud-Computing, presented at the INFORMS meeting in Dallas in 1997. He said:

"*Cloud computing is a computing paradigm where the boundaries of computing will be determined by economic rationale rather than technical limits alone.*" (Ramnath Chilapa, [393])

In 2008, the IEEE published an article by Carl Hewitt which defines cloud computing as:

"*A paradigm in which information is permanently stored in servers on the Internet and cached temporarily on clients that include desktops, entertainment centers, tablet computers, notebooks, wall computers, handhelds, sensors, monitors, etc.*" (Carl Hewitt [185])

Brin J.S Chee and Curtis Frankin, Jr, in their book published in 2010, defined cloud computing is as follows:

"*Cloud computing is an information-processing model in which centrally-administered computing capabilities are delivered as services, on an as-needed basis, across the network to a variety of user-facing devices.*" (Brin & Curtis [95])

The formal definition of cloud computing comes from the National Institute of Standards and Technology (NIST) in 2011:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models." (NIST, [280])

As a summary of this section, we can say that:

"*Cloud computing is a dynamic distributed computing paradigm based on the consumption of computing resources (servers, storage, applications, bandwidth, etc.) on-demand through Internet technologies for the largest possible number of users.*"

1.3.3 Five-Four-Three principles of cloud computing

In light of NIST's definition, cloud computing relies on *(i)* five fundamental characteristics that promote cloud computing, *(ii)* four deployment models used to describe cloud computing opportunities, and *(iii)* three basic service offering models (see Figure 1.8).

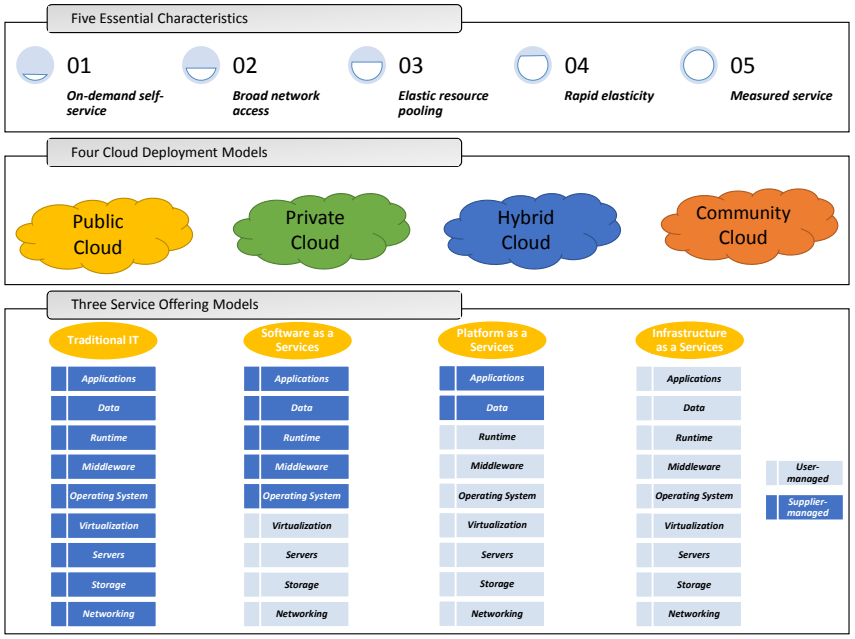


FIGURE 1.8: The NIST definition of cloud computing

1.3.3.1 Five essential characteristics

The following features should be included in cloud computing services:

- *On-demand self-service*: cloud computing allows users to consume services such as servers, storage or computing capacity, development platform, etc., according to their needs and automatically without requiring human interaction with the service provider.
- *Broad network access*: cloud computing delivers all end-user services remotely over the network and at any time through standard mechanisms supporting heterogeneous client platforms such as mobile devices and desktops.
- *Elastic resource pooling*: using the automatic management system of cloud computing, resources are pooled to serve multiple users using a multi-tenant model. Different physical and virtual resources are assigned and dynamically reallocated based on user demand without knowing the precise location of the resources, provided without their intervention for control.
- *Rapid elasticity*: differing user workloads cause significant changes in the volume of resources requested. Cloud computing can increase or decrease the computing resources provided immediately and automatically on demand to adapt to these changes.
- *Measured service*: like public services such as electricity, water, gas, etc., cloud computing has accounting mechanisms based on the principle of pay-as-you-use due to the possibility of measuring the resources and services used by end-users.

1.3.3.2 Four cloud deployment models

The four models of cloud computing deployment are described below:

- *Public cloud*: this model is the most popular form of cloud computing. It is the computer version of the "utility" model that we all use to consume electricity or water in our homes [189]. In simple terms, it is an extensive IT infrastructure that provides several resources and services like apps, servers, and storage, to the general public on-demand via the Internet. It may be owned, administered, and managed by a company, an academic institution or a government organisation, or a combination of these [89]. However, this deployment model lacks precise control over data, networks, and security settings, hindering their effectiveness in many business scenarios [505]. The leading public clouds include Amazon Web Services (AWS)[2], IBM Bluemix[3], Google cloud Platform[4], or Microsoft Azure[5].
- *Private cloud*: as its name suggests, this model is not accessible to the general public. A private cloud is a set of private resources and services provided to a specific consumer or organisation comprising multiple consumers, effectively controlling data, security, and quality of service. This model's main characteristics reside in the integration, data, and

[2]Amazon Web Services (AWS): https://aws.amazon.com/fr/what-is-aws/?nc1=f_cc, Accessed: July. 2020.
[3]Bluemix: https://www.ibm.com/blogs/cloud-computing/2014/04/21/ibm-codename-bluemix/, Accessed: July. 2020.
[4]Google Cloud Platform: https://cloud.google.com/, Accessed: July. 2020.
[5]Microsoft Azure: https://azure.microsoft.com/en-us/?b=16.26, Accessed: July. 2020.

critical application security solutions. Some examples are Amazon VPC[6], Eucalyptus[7], OpenStack[8], and VMWare[9].

- *Community cloud*: is another important cloud model that is in use today. A community cloud is a form of public cloud limited to a particular community, generally professional, with a common interest. Their advantage is that they allow several independent entities to obtain a shared non-public cloud's financial benefits while avoiding security and regulatory issues.

- *Hybrid cloud*: the hybrid cloud is a natural evolution of the deployment models discussed above. It can take any combination of the three forms - public, private, and community. The hybrid cloud inherits the private cloud's security features and can use the public cloud resources for more efficiency. Some examples are RightScale or Flexera[10], Asigra Hybrid Cloud Backup[11], and QST[12].

1.3.3.3 Three service offering models

According to Figure 1.8, cloud computing comprises three basic types of services: *(i)* software as a service (SaaS), *(ii)* platform as a service (PaaS), and *(iii)* infrastructure as a service (IaaS). Differ from each other for the level of abstraction, task automation, and how the user accesses IT resources. These levels of abstraction reduce the effort required by the service consumer to create and deploy systems [281].

- *Cloud SaaS*: the SaaS type of cloud computing services correspond to a model where hardware, hosting, application framework, and software are dematerialised and offered on-demand in the form of services through web portals. Customers can take these services on a pay-per-use basis. In this type of service, the provider takes care of the management and control of the necessary resources, and the customer does nothing except a few very specific configurations. The best-known example of cloud SaaS is Microsoft Office 365[13] or also Google Docs[14].

- *Cloud PaaS*: PaaS is simply a packaged, ready-to-use development or operating framework. Unlike cloud SaaS, cloud PaaS gives the consumer the freedom to control the applications deployed and possibly the application-hosting environment's configuration settings, but does not manage or control the underlying cloud infrastructure. Major PaaS cloud service providers are Google App Engine (GAE)[15], Salesforce[16], and Microsoft Azure.

- *Cloud IaaS*: IaaS gives customers the same features as PaaS, besides being fully responsible for controlling the leased infrastructure. The IaaS client is responsible for all aspects of

[6] Amazon VPC: https://aws.amazon.com/vpc/, Accessed July. 2020.
[7] Eucalyptus: https://www.eucalyptus.cloud/, Accessed July. 2020.
[8] OpenStack: https://www.openstack.org/, Accessed July. 2020.
[9] VMWare: https://www.vmware.com/company.html, Accessed July. 2020.
[10] Flexera https://www.flexera.com/products/agility/cloud-management-platform.html, Accessed July. 2020.
[11] Asigra Hybrid Cloud Backup https://www.asigra.com/cloud-saas-solution, Accessed July. 2020.
[12] QTS: https://www.qtsdatacenters.com/, Accessed July. 2020.
[13] Microsoft Office 365: https://www.microsoft.com/en-us/microsoft-365, Accessed July. 2020.
[14] Google Docs: https://www.google.com/docs/about/, Accessed July. 2020.
[15] Google App Engine: https://cloud.google.com/appengine, Accessed Juy.2020.
[16] Salesforce: https://www.salesforce.com/fr/products/what-is-salesforce/, Accessed July.2020.

the security of the system they use, except physical security. Amazon Web Services[17] and Mosso/Rackspace[18] are examples of cloud IaaS providers.

- ❏ *Cloud XaaS*: XaaS (or Everything-as-a-Service) represents nothing less than cloud computing's future to cover all IoT-specific computing resources [156]. This model, invented by IBM in 2014, brings together all the technologies delivered via the cloud mentioned above. In general, XaaS refers to the infrastructure, platform, software, networking, caching or storage and many other types of resources mentioned in cloud services [91].

1.3.4 Cloud computing architecture

Cloud computing technology, like any technological model, has an architecture that describes how it works. As mentioned in [22, 198, 505], and as shown in Figure 1.9, cloud architecture is divided into four layers: the Hardware layer, the Infrastructure layer, the Platform layer, and the Application layer. In the following, we explain each of them in brief:

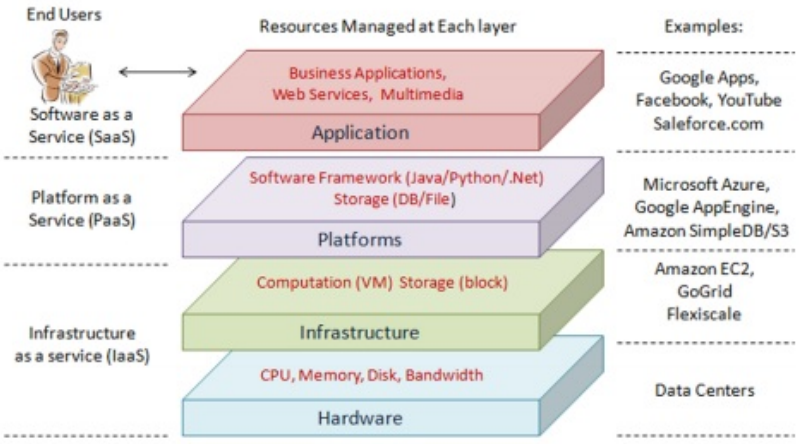


FIGURE 1.9: Cloud computing architecture (from [505])

- ❏ *Hardware Layer*: This layer (also known as the Physical layer) includes cooling systems, power, in addition to routers and switches to connect thousands of physical servers and other devices that can be physically managed and controlled [198].
- ❏ *Infrastructure Layer*: The infrastructure layer or virtualisation layer includes storage facilities, virtualised servers, and networking. This layer has many major advantages, such as dynamic resource partitioning and assignment using virtualisation technologies such as KVM (for Kernel-based Virtual Machines)[19], VMware[20], and Xen[21] [505]. The services offered in this layer can be classified into IT resources, data storage, and communications [22].
- ❏ *Platform Layer*: This layer includes services such as operating systems, application frameworks, and requisition structures. The purpose of the platform layer is to minimise the burden of deploying applications directly into VM containers [505].

[17] Amazon Web Services: https://aws.amazon.com/what-is-aws/?nc1=f_cc, Accessed July. 2020.
[18] Mosso/Rackspace: https://www.rackspace.com/cloud, Accessed July. 2020.
[19] KVM: http://www.linux-kvm.org/page/Main_Page, Accessed July. 2020.
[20] VMware: https://www.vmware.com/products/esxi-and-esx.html, Accessed July. 2020.
[21] Xen: https://www.citrix.com/products/citrix-hypervisor/, Accessed July. 2020.

- ❏ *Application Layer*: The application layer mainly includes software systems provided as a service. This layer is the layer with which the end-users interact directly to obtain these SaaS [198].

1.3.5 Advantage & challenges of cloud computing

Cloud computing is usually associated with a variety of advantages. Here we mention their importance:

- ✔ One of the essential advantages of cloud computing is the *accessibility* of services anytime, anywhere, from any connected terminal making users' lives much easier.
- ✔ Another advantage is the *availability* of virtually unlimited resources. Due to the high availability and disaster recovery system of cloud computing, the availability of data and services to consumers in the event of an outage can be guaranteed immediately.
- ✔ With the *pay-as-you-go* policy supported by the cloud management system, it is easy to access the resources delivered as services with minimal upfront costs. This model allows the customer to avoid investing in expensive computer equipment that is rarely used.
- ✔ The ability to increase or decrease the need for computing resources provided by the cloud makes it attractive to many consumers, especially to commercial enterprises. This is known as *scalability*. This scalability is achieved quickly and easily with no serious issues at stake. This is known as *flexibility*.

At present, these benefits remain theoretical, given the nature of the concept itself. Future studies will determine the role that cloud computing will play in business innovation, the development of the IoT, and our daily lives.

In the rest of the section, we briefly present the challenges of cloud computing.

- ❏ Cloud computing is a platform for providing unlimited resources to users over the Internet. In this case, the first challenge we observe is connectivity. If we lose the Internet connection, we lose the link to the cloud and, consequently, the data and applications.
- ❏ The biggest challenge of cloud computing is security and privacy, which are all cloud customers' main concerns. This issue became the main topic of preoccupation in 2018, where more than 70% of respondents indicated it in the referenced survey [133]. At present, security has improved but is not yet ideal.
- ❏ The other issue is that a user may lose control of the hardware and data. While many consumers like to control their data, naturally, losing control is a problem for not choosing these services. Therefore, reduce the reliability and efficiency rate in the cloud.
- ❏ The location of data storage is another challenge in cloud systems. To ensure a high level of service availability to customers, cloud service providers deploy sufficient redundancy in geographically distributed locations. Unfortunately, most countries have explicit laws that require data to be stored within the country because of government laws in some states that do not respect the confidentiality of personal information, such as the PATRIOT Act in the United States, which stipulates that the U.S. government can access all data stored in the United States.

1.4 Fog computing

1.4.1 Limitations of cloud computing in the IoT

It is clear that the IoT arrived and continued to grow until it was successfully combined with cloud computing. This integration is known as "Cloud of Things" or "CoT" and is one of the main research trends to address IoT's challenges. Specifically, it helps manage IoT resources and provides a more cost-effective, simplifying data processing flow [2, 38, 39]. Despite the benefits brought by the use of the CoT paradigm, it also introduces new challenges in addition to the challenges mentioned in subsection 1.3.5 for the IoT system that cannot be solved by cloud computing, such as network bandwidth constraints, strict latency requirements, uninterrupted services with intermittent connectivity, resource-constrained devices and wide distribution of devices [6, 38, 39, 66, 80, 103, 103, 119, 273, 514].

- ***Bandwidth***: A large number of IoT devices send large and high-frequency data continuously to the cloud for analysis. Sending all this data to the cloud will necessarily surpass network bandwidth availability [103, 265], becoming a bottleneck for the cloud [6]. As a result, about 90% of the data generated by IoT devices is stored and processed locally and not in the cloud, according to ABI Research [213].

> *"... and of the captured volume, on average over 90 percent is stored or processed locally without a cloud element..."*
>
> *Rhea Kelly from ABI Research*

- ***Latency***: Many IoT applications, such as smart grid systems, autonomous vehicle networks, video surveillance systems, and health applications require low latency (a few milliseconds). These requirements are outside the capabilities of cloud services due to the large distance between IoT devices and the cloud, where the number of hops the package needs to make to travel and getting the response back is large and may not be desired [169].
- ***Uninterrupted***: In the last-mentioned systems, unstable and intermittent network connectivity with cloud services is often the top priority. It is not easy to provide it because of the long distance between the distant cloud and IoT devices [80]. As a result, a new computing and network architecture will be required to mitigate or solve the problem [103].
- ***Resource-constrained***: Several IoT devices have limited resources in terms of memory, processing, and energy. With these limited resources, the IoT devices will not fulfil all their computing needs or interact directly with the cloud without including at least some powerful hardware as an intermediate to connect to the cloud because these interactions often require intensive processing and complex protocols [103, 119].
- ***Geographical distribution***: In IoT, connected devices are deployed in geographically distributed locations. As these devices have grown in recent years, their space has become very large geographical areas [482]. This large-scale distribution makes it very difficult to find a cloud infrastructure location to meet all the requirements of IoT applications [119].

The challenges that the centralised cloud paradigm faces are outlined in the above review. The following sections will look at the various post-cloud paradigms proposed by multiple organisations in recent years. These paradigms are based on the concept of extending cloud computing from the data centre to where things are, addressing network bottlenecks and improving the processing speeds and efficiencies of customer services.

1.4.2 Post-cloud computing

The post-cloud computing phase is known as the decentralisation phase. During this phase, the academic and industrial sectors conducted several studies to find new solutions for overcoming cloud computing limitations discussed in the previous section. According to the surveys [119, 295, 320, 514], several similar computing paradigms are proposed besides *fog computing*, such as *edge*, *cloudlet*, *dew*, and *mist computing*. This section briefly describes the origin and main idea of these paradigms, while clarifying the differences between them.

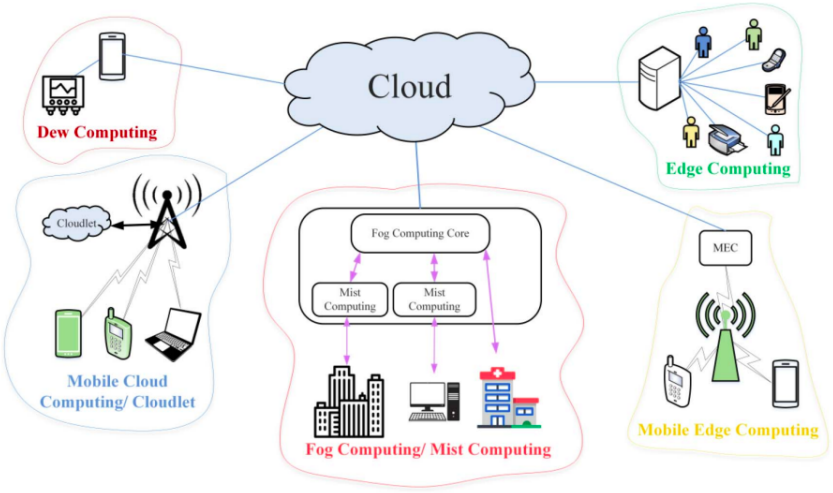


FIGURE 1.10: Post-cloud computing paradigms (from [310])

1.4.2.1 Edge computing

Edge computing is a paradigm whose philosophy in the literature preceded cloud computing. However, interest in this in academic and industrial sectors began with the development of the IoT and its new challenges. The first article mentioning the term "computing at the edge" was published in 2004 [339], while "edge computing" was introduced by Karim Arabi during a conference keynote in 2014 [31] and then in an invited talk at MIT's MTL Seminar in 2015 [32].

> *"Edge computing refers to the enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of IoT services."*
>
> *Shi et al. [380]*

The main principle of edge computing is to push applications, data, and services away from the cloud to the edge of the Internet to address the centralised cloud system issue. In this paradigm,

the end devices act as not only data consumers but also data producers. These devices request services from the cloud and provide services (processing, storage, caching, and load balancing) to the cloud, instead of just requesting from the cloud traditionally [381].

1.4.2.2 Fog computing

The most popular post-cloud paradigm is fog computing. Flavio Bonomi first proposed this concept, Vice President of Cisco Systems, in 2011 and was explained in detail in 2012 [65, 66, 429]. The main objective of this paradigm is the ability to manage large numbers of IoT devices and large volumes of data for real-time, low-latency applications, delivering better and faster processing and application services.

> *"FOG (From cOre to edGe) computing is a distributed computing paradigm born of necessity to move the services of cloud computing (compute, storage and networking) at the edge devices. Edge devices in this context are referred to as "Things" which include a wide variety of sensors, actuators, mobile devices, embedded systems, etc."*
>
> *Vaquero et al. [429]*

Pan [320] explain fog computing's principle as migration of cloud computing and its services to devices such as routers, routing switches, multiplexers, and more located at the network's edge. Fog computing facilitates location awareness, scalability, and interoperability [310]. Also, fog computing supports widespread geographical distribution networks, mobility, content distribution, and heterogeneity [66, 365].

1.4.2.3 Mist computing

Cisco also proposed the mist computing paradigm. It is a small form of fog computing that locates at the edge by pushing some computation to the edge of the network, actuator devices, and the sensor, which has built the network for the cloud data centre. The devices in mist computing are usually microcomputers and microcontrollers that send data to the fog nodes and, in some cases, directly to the cloud. As shown in Figure 1.10, mist computing is not considered a mandatory layer of fog computing [192].

> *"Mist computing is a lightweight and rudimentary form of fog computing that resides directly within the network fabric at the edge of the network fabric, bringing the fog computing layer closer to the smart end-devices. Mist computing uses microcomputers and microcontrollers to feed into fog computing nodes and potentially onward towards the centralized (cloud) computing services."*
>
> *Iorga et al. [192]*

The mist computing paradigm has decreased the latency, increased the throughput rate, and increased the autonomy of a solution [423]. Cloud, fog, and mist computing are complementary to each other, where the application tasks, which are more computationally intensive and less delay-sensitive tasks, can be executed in the gateway of the fog nodes. In contrast, the less

computationally intensive and more delay-sensitive tasks can be executed in the edge devices. The processing and the collecting of data are still performed and stored in the cloud computing [310].

1.4.2.4 Cloudlet computing

IoT devices require multiple network hops to access the cloud, resulting in high response times and low bandwidth. In 2009, Mahadev Satyanarayanan et al. [367] introduced the term cloudlet as a new vision of mobile computing for mobile devices that free it from severe resource constraints by enabling resource-intensive applications to leverage cloud computing free of WAN delays, jitter, congestion and failures.

> *"A cloudlet is a trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and is available for use by nearby mobile devices"*
>
> *Satyanarayanan et al. [367]*

Cloudlet is a small, trusted cloud infrastructure that promotes small-scale cloud data centres at the edge of the network, close to a mobile device but not on the mobile device. In the cloud-to-things continuum hierarchy, the cloudlet is the intermediate level: mobile device — cloudlet — cloud (see Figure 1.10).

1.4.2.5 Dew computing

Dew computing is a new structural layer of distributed computing hierarchy was introduced in 2012 and explained their basic architecture in detail in 2015 [387, 444, 445, 447].

> *"Dew computing is an on-premises computer software hardware organization paradigm in the cloud computing environment where the on-premises computer provides functionality that is independent of cloud services and is also collaborative with cloud services. The goal of Dew computing is to fully realize the potentials of on-premises computers and cloud services."*
>
> *Yingwei Wang [446]*

Dew computing [444] is situated at the ground level of the cloud and fog computing environment [387]. This paradigm uses on-premises computers to provide functionality independent of cloud services and collaborate with cloud services [447]. The main idea of Dew computation is to maximise resource utilisation before processing is transferred to the cloud computing server, reduce the complexity and improve the productivity of scalable distributed computing.

1.4.2.6 Simple comparative study

Considering some researcher surveys [187, 310, 338, 348, 367, 514], we have summarised the differences between cloud computing and post-cloud computing in Table 1.2. Then we compare the different paradigms of post-cloud computing in Table 1.3.

Table 1.2: Simple comparison of cloud computing and post-cloud computing

	Cloud computing	Post-cloud computing
Latency	high	low
Access Network	wired or wireless	mainly wireless
Mobility support	limited	supported
Architecture	centralized	hierarchical, and distributed
Computation and storage capabilities	strong	weak temporarily
Execution place of computation	large data centre	network edge, adjacent device, device itself
Distance to users	far from users (multiple network hops)	physically close to users (single network hop or few network hops)
Number of server nodes	few	large
Hardware resources	rich and extensible storage space and computing capacities	limited storage and computing capacities
Geo-distribution	centralised	distributed
Geographic coverage	global	local area or wider
Bandwidth costs	high	low
Types of services	information collected from the global scope	local information services for a particular deployment environment
Location of service	within the Internet	at the edge of local networks
Energy consumption	high	low

Table 1.3: Detailed comparison of typical post-cloud computing paradigms

	Cloudlet	Fog	Edge	Dew	Mist
Distance to user	close	close	very close	very close	very close
Access Mechanisms	Wi-Fi and Mobile Network	Wi-Fi and Mobile Network	Bluetooth, Wi-Fi, ZigBee, etc.	Ad-hoc	Mobile Network, Bluetooth, Wi-Fi, ZigBee, etc.
Latency	low	medium	low	low	low
Bandwidth	low	low	low	low	low
Mobility	supported	supported	supported	supported	supported
Distance between client and server	one or multiple hops	one or multiple hops	one hop	one hop	one hop

	Cloudlet	Fog	Edge	Dew	Mist
Deployed hardware	data centre in a box	routers, switches, access points, gateways	servers running in base stations	sensors and smartphones	microcomputers, microcontrollers, and microchips
Usage of end devices	no	yes	no	yes	yes
Number of users/devices	thousands	thousands to millions	hundreds to thousands	few devices	hundreds
Context awareness	low	medium	high	high	high
Location awareness	yes	yes	yes	yes	yes

Based on the comparison and analysis of the characteristics and evolution of these computing paradigms, we can conclude that post-cloud computing not only extends to the edge of the network it also differs fundamentally from cloud computing in its architecture, storage computing and execution, and other aspects. Post-cloud computing acts as a bridge between IoT devices and large-scale cloud computing. Therefore, post-cloud computing supports large-scale sensor networks and real-time interactions between IoT devices, allowing reduced latency and time-sensitive IoT applications.

Table 1.3 shows that although these computing models' names differ, they have similar features, including low latency and close to end-users, and unique characteristics. Fog computing emphasises the infrastructure between the edge devices and the cloud platform, while edge computing focuses on the infrastructure that fog computing cared about and the wireless access network from the point of view of communication operators. The main focus of the cloudlet is on providing real-time services to a large number of bandwidth-constrained mobile applications, while Dew computing emphasises the collaboration between the IoT devices and cloud computing, regardless of the edge and the wireless access network. In general, the post-cloud paradigms push more applications and services from the cloud to the edge network, reducing the data transfer time and congestion of the network and effectively meeting the demands of real-time or latency-sensitive applications.

1.4.3 Fog computing definition

Various researchers in the literature have defined fog computing in different ways. In this section, we will cite a few basic definitions.

Yi et al. [485] have provided a general definition of fog computing. In this definition, the authors defined fog computing as an additional resource-rich layer between end-devices and the cloud to address the challenges mentioned above.

"*Fog computing is a geographically distributed computing architecture with a resource pool which consists of one or more ubiquitously connected heterogeneous devices (including edge devices) at the edge of a network and not exclusively seamlessly backed by cloud services, to collaboratively provide elastic computation, storage, and communication (and many other new services and tasks) in isolated environments to a large scale of clients in proximity.*" (Yi et al., [485])

Vaquero and Rodero-Merino in [429] provided a clear definition of fog computing. They focused on the essential characteristics that distinguish it from other related computing models: ubiquity, improved network capabilities as a hosting environment, and better support for cooperation between devices.

"*Fog computing is a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralized devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third parties. These tasks can be for supporting basic network functions or new services and applications that run in a sand-boxed environment. Users leasing part of their devices to host these services get incentives for doing so.*" (Vaquero & Rodero-Merino, [429])

The IBM definition [44] identifies fog computing and edge computing as the same computing paradigm. According to Qiu et al. [338], fog computing focuses on the infrastructure between the edge devices and the cloud platform, while edge computing focuses not only on the infrastructure that fog computing cared about but also on the edge devices with enormous quantities. On that basis, he said: "... fog computing is a subset of edge computing...".

"The term fog computing or edge computing means that rather than hosting and working from a centralized cloud, fog systems operate on network ends. It is a term for placing some processes and resources at the edge of the cloud, instead of establishing channels for cloud storage and utilization." (IBM, [44])

There is another definition given by Neha et al. in their survey [295], and they have defined fog computing as all devices having a computing and storage capacity.

"*Fog computing is a distributed computing platform where most of the processing will be done by virtualized and non-virtualized end or edge devices. It is also associated with the cloud for non-latency-aware processing and long-term storage of useful data by residing in between users and the cloud.*" (Neha et al., [295])

NIST Special Publication 800-191 [191] defines it as a horizontal, physical or virtual resource paradigm that resides between edge devices and cloud computing.

"*Fog computing is a horizontal, physical or virtual resource paradigm that resides between smart end-devices and traditional cloud or data centers. This paradigm supports vertically-isolated, latency-sensitive applications by providing ubiquitous, scalable, layered, federated, and distributed computing, storage, and network connectivity.*" (NIST, [191])

1.4.4 Architectures of fog computing: High-level overview

Fog computing is a new paradigm created due to the need to deploy computing infrastructure on the edge of the network, which has resulted from the convergence of cloud computing and the IoT. In the literature, several studies have been carried out to establish fog architecture's layered concept [52, 119, 187, 191, 245, 295, 402, 485]. However, to date, no standard architecture is available.

According to the study of NAHA et al. [295], many architectures have been proposed, including three-layer architecture (Figure 1.11), four [33], five [117], six [3] and seven layers [295]. Every proposed architecture has its justifications for its claims. After reviewing the mentioned literature, we find that the three-layer architecture is the reference fog computing architecture if we ignore the user's claims. Therefore, in this part, we present fog computing's three-layer architecture to illustrate fog computing in the cloud-to-things continuum.

As depicted in Figure 1.11, a fog computing architecture is composed of: things/end-devices, fog, and cloud layer.

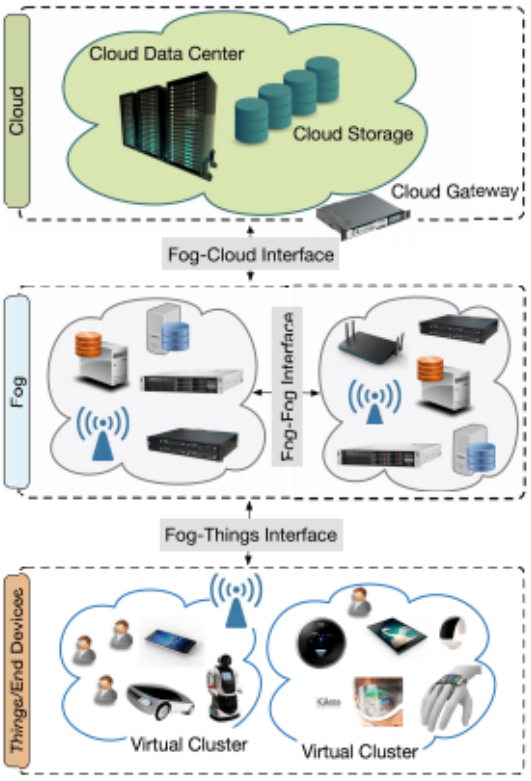


FIGURE 1.11: High level fog computing architecture (from [292])

- **Things layer:** At the lower levels are the end-devices responsible for detecting, collecting, and sending the data to the top layer for storage or processing. Sometimes, end-devices with significant computing capabilities like smartphones can also do local processing before including the upper layers.
- **Fog layer:** The next layer is the essential layer of the architecture and is the fog layer. This layer consists of all intermediate computing devices called fog nodes or servers. Fog nodes can be set-top-boxes, access points, roadside units, cellular base stations, gateways, routers, etc., located anywhere between cloud and end devices. These fog nodes can collaboratively share storage and computing facilities [292].
- **Cloud layer:** The cloud constitutes the highest layer, offering a large pool of resources at a low cost without latency guarantees. It comprises several servers with high computational and storage capabilities and provides different services (discussed in section 1.3). Differently from the traditional cloud computing architecture, in the fog architecture, some computation, or services might be proficiently moved from cloud to fog layer to reduce the load on cloud resources and increase the efficiency [119].

According to [103, 292], Figure 1.11, and Table 1.4, the fog architecture has three main interfaces, namely:

- ❖ **Fog-Cloud interface:** Fog-Cloud interface was introduced to support fog and cloud computing collaborations to provide end-to-end services, including how the cloud will distribute service to the fog.
- ❖ **Fog-Fog interface:** The Fog-Fog Interface is the interface that supports collaboration between multiple fog nodes to share on-demand services such as data storage and computing tasks for one or several users.
- ❖ **Fog-Thing interface:** Devices or end-users can easily, efficiently, and securely access fog resources and services through the Fog-Thing interface. End-devices and fog layer communicate through Local Area Network (LAN) while communication between the end-devices and cloud layer can be done either through the fog or without fog over the Wide Area Network (WAN) [288].

In 2017, Openfog Consortium [112] defined a new architecture extended of the three-layer called N-tier architecture. This architecture has the same three entities as the three-layer architecture. The fog layer in this architecture is composed of several levels of fog nodes (N-tiers) and, as the nodes move further away from the end devices, they gain more computing power and intelligence, in addition to the ability to communicate on both sides horizontally and vertically.

TABLE 1.4: Interfacing of Fog to Cloud, Fog to Fog, Fog to IoT

Interface	The role it plays
Fog-Cloud interface	- It is considered compulsory to support collaboration between fog to cloud and cloud to fog services -It also supports functionalities like supervision at the fog -It also transfers data for processing, comparing between each other
Fog-Fog interface	- It must have a pool of resources to support processing with each other -Multiple fog nodes act together to support backups for each other -All the fog nodes deployed can share their data, computation, and processing for one or several applications
Fog-Things interface	- Fog computing provides services to widely distributed IoT devices like smart devices and sensors - It provides IoT devices to access fog services in a user-friendly environment - Fog computing provides high QoS to the IoT devices or end-users

1.4.5 Salient features of the fog computing

According to [39, 91, 119, 169, 196, 265, 295, 343], fog computing has many characteristics that make it an attractive technology. These characteristics are summarised as follows:

- ✓ **Geographical distribution:** Unlike cloud computing, fog computing is not centralised. This paradigm is characterised by deploying its nodes in multiple locations to deliver distributed services and applications everywhere.

- ✓ **Very large number of nodes:** As a consequence of the wide geo-distribution, it is evident that this paradigm is composed of a large number of nodes deployed in distributed geographic areas to provide services to mobile devices in those areas.
- ✓ **Edge location:** According to the definitions given in section 1.4.3, the fog nodes are located close to end-users or end-devices.
- ✓ **Heterogeneity:** Fog nodes have many different forms and are deployed in a variety of distributed environments. Fog computing ensures that these heterogeneous devices can communicate effectively to collect data from multiple sources.
- ✓ **Scalability:** Fog computing is a distributed multi-layered environment that can easily support a growing number of terminals. Through intercommunication between fog nodes, the scalability of the services they provide is improved.
- ✓ **Interoperability and federation:** Seamless support of certain services like real-time services requires the cooperation of different providers. Therefore, fog computing can interact and standardise services between domains.
- ✓ **Reliability:** Fog computing is more reliable than centralised computing paradigms because one node's failure does not affect the entire system.
- ✓ **Location awareness:** The large distribution and communication between fog nodes allow us to understand their logical location in the context of global systems and to become aware of the environment.

1.4.6 Essential benefits of fog computing

The movement of computational load to fog networking from the cloud enables IoT structure to develop for diverse applications and services. Fog computing advantages are as follows:

- ✎ **Reduction of network traffic:** As shown in Figure 1.11, the fog layer is the intermediate layer between the data generators representing the IoT devices layer and the cloud layer. Unlike centralised cloud computing architecture, fog computing allows data to be collected, processed and stored from IoT devices in multiple distributed fog nodes, reducing the cloud's load and network traffic.
- ✎ **Low latency:** Due to the proximity of the fog nodes to the IoT devices and their cognition, the services provided by fog computing (computing, storage, communication [343]) are delivered in a much quicker time frame, unlike the cloud, which has a much higher latency [191].
- ✎ **Increasing data quality:** Quality of data would be increased by eliminating low-quality data such as repetitive, corrupted, or noisy data and the integration of received data in a fog layer [91].
- ✎ **Increasing efficiency:** Local processing of data and eliminating corrupted, repetitive, or unneeded data in the fog layer reduces the network load and increases network efficiency. Because the transferred data to the cloud must be processed, stored, and analysed in the cloud, by decreasing the amount of data, cloud processing and storage needs would also decrease [91].

- ✎ **Real-time Interactions and Analytics:** Another advantage closely related to the proximity of distributed nodes to terminals, cognition and ultra-low latency of fog computing is its interactions and analysis in real-time, in contrast to the cloud, which has a much higher latency.
- ✎ **Predominance of wireless access:** One of the main goals of the IoT is to enable energy-efficient access to large-scale wireless devices that require distributed analysis and communication [480]. For this reason, fog computing is very well suited for IoT wireless access networks due to the wide distribution, reliability and scalability of the fog nodes [191].
- ✎ **Support mobility:** Fog computing enables mobile devices to move from one location to another without interrupting fog-compatible services [480]. The latter results from the location-awareness, large-scale geographic distribution, real-time interaction, and the predominance of wireless access mentioned above.
- ✎ **Reduced costs:** Local data processing without sending to cloud computing reduces network bandwidth, cloud processing, storage, and energy.
- ✎ **Increasing the level of privacy:** In IoT systems, sensors may generate and transfer sensitive and confidential data, but transferring them without manipulation and encryption bears the risk of disclosure. Devices cannot handle complicated mathematical operations—the privacy-preserving mechanisms, such as encrypting algorithms in end-devices, maybe impossible. Therefore, privacy, data manipulations and encryption algorithms can be done in a fog layer. Nevertheless, the protection of fog devices is another issue that will be investigated further [91].

Several other advantages and characteristics of fog computing are discussed in the literature, often with a different outline [100, 187, 266, 305, 485]. Nevertheless, we believe that the advantages presented in this section are generic enough to be considered the key concepts from which the other features derive.

1.5 Conclusion

In this chapter, we have covered almost all aspects of our work context for our thesis. We first discussed the concept of the IoT, its definition, its architecture and the main applications that use it as infrastructure in section 1.2. Next, in section 1.3 and 1.4, we have reviewed the different modern paradigms (cloud, cloudlet, fog, and mist computing) that have emerged to support the deployment of applications based on the IoT.

CHAPTER 2

INTELLIGENT VIDEO SURVEILLANCE SYSTEMS (IVSS)

Chapter contents

2.1 Introduction

The video surveillance system, more commonly known as closed-circuit television (CCTV), is part of the physical security sector, which consists of remotely monitoring objects or places (public or private) using cameras that transmit the videos to the surveillance base station (BS) for recording and playback on a screen in the control office [161]. Many cyber-physical systems have employed CCTV systems, including traffic monitoring, to measure traffic flow [142] and to detecting incidents on roads or highways [319]. Other applications include health care [333, 333], public safety [308], wildlife monitoring [307], intelligent buildings [247], industrial automation [369], and environmental and weather monitoring [41, 401].

Formally, the CCTV systems are composed of a set of cameras connected to control screens. These systems give a global view of a large area to a limited number of qualified operators. Their mission is to discover abnormal situations, and, depending on the seriousness of the situation, they can take the necessary measures. As the number of cameras increases, this mission becomes more complex and may become impossible. For this reason, research has sought to automate the video surveillance process.

During the last decade, two particular aspects of artificial intelligence have received attention and have improved considerably. On the one hand, the development of artificial vision algorithms. This latter can process complex visual data and provide human-like feedback on the scene's evolution under surveillance. Because of this evolution, traditional surveillance systems have been replaced by new, more innovative surveillance systems, known as Intelligent Video Surveillance Systems (IVSS). IVSS systems can automatically identify abnormal behaviours and patterns in a video thanks to the introduction of artificial intelligence, paternity recognition and computer vision technologies. On the other hand, advances in distributed computing and distributed intelligence. These allow devices to be treated as distinct entities, capable of adapting to the evolution of the scene and the complexity of the communication network, reducing the bandwidth of information and deriving a better interpretation of people's dynamics and objects moving in the scene.

In this chapter, we will present state-of-the-art on video surveillance and its evolution. In section 2.2, we will introduce some reminders on the evolution of the video surveillance system. Section 2.3 will present an overview of the different components of a video surveillance system, the different image processing techniques used, and a study on the different classifications of

this system and its challenges for the next generation. Next, in section 2.4, we will describe the motivation for using distributed intelligence and a cooperative distributed system in video surveillance systems. Before concluding this chapter, we will present in section 2.5 a predictive view of the next generation of the intelligent video surveillance system.

2.2 Review of video surveillance system evolution

When CCTV systems have proven their benefits for much physical security in public spaces, the interest in developing video surveillance applications has increased considerably in the industrial and academic world. CCTV systems have gone through several years of evolution until now to enhance their performance and improve their service quality (QoS). This section presents the different steps that the video surveillance system has gone through to date. Table 2.1 shows an analytical overview of these steps.

2.2.1 First-generation surveillance systems: All analog (1960-1980)

The first use of CCTV systems dated back to the 1960s and continued until the 1980s. At that time, the systems used analogue technologies for data acquisition and transmission. This operating cycle begins with acquiring videos/images from cameras directly connected via coaxial cables with a monitor and recording equipment. The received data is stored on cassettes and displayed on screens at the control office level, allowing human operators to observe the environment and analyse the scenes. The main challenges of this generation are **1)** Data quality degradation resulting from converting a digital image to an analogue composite video signal. **2)** The need for very high bandwidth for analogue data transmission. **3)** The increase of data required the need to adopt an efficient and flexible storage mechanism in addition to the storage capacity itself. **4)** The difficulty of controlling all the scenes manually to find the targets by human operators. It should be noted that as the number of cameras increases, this mission becomes impossible.

2.2.2 Second-generation surveillance systems: Hybrid system (1980-2000)

The second generation begins in the years 1980 to 2000. In this generation, the systems have advantages over the first generation due to analogue and digital technologies' hybridisation. Due to data digitisation, the degradation of data quality has improved significantly. It has been compressed and stored on hard drives using digital recorders or transmitted over a network for remote monitoring.

During this generation, research has mainly focused on developing new systems that introduce a certain level of automation and local intelligence, such as automatic event detection, to help and facilitate human operators to monitor a large area equipped with several cameras.

2.2.3 Third-generation surveillance systems: All digital (2000-present)

The third generation of video surveillance systems starts from the year 2000 until now. The systems of this generation are based on digital technologies, from cameras to data processing.

All transmissions are via the Internet. Thanks to this significant transformation, the CCTV system has become autonomous and more intelligent. It can provide reliable real-time surveillance through advanced video analysis and artificial intelligence techniques without human intervention. The following are the essential modules for intelligent video surveillance.

- Digital video acquisition;
- Preprocessing at camera level;
- Data compression;
- Data transmission;
- Data processing;
- Storage and display.

In this generation, most research focuses on developing a more intelligent and fully autonomous video surveillance system capable of managing complex real-time events and manipulating a large number of cameras to cover large areas, the distribution of processing capabilities, and the combination of various heterogeneous cameras.

2.3 Intelligent video surveillance systems (IVSS): Fundamental concepts

2.3.1 Domain definition

Before elaborating on the subject of this section, we first present definitions of the terminologies and technical concepts to highlight the points of their intersections and their differences.

2.3.1.1 Surveillance

Due to the rapid development of video surveillance, many definitions do not capture surveillance's new conceptions. Today, many of the surveillance systems are not explicitly introduced to "a suspected person". It also applies to different contexts, including, but not limited to: places and geographical areas, networks, systems, and classes of people. In the following, we present some definitions of the surveillance concept.

David Lyon suggests that surveillance is any collection and processing of personal data, whether identifiable or not, to influence or manage those whose data have been garnered [261].

Ross Bellaby writes that 'surveillance' can cover a wide range of activities, from CCTV cameras and 'covert surveillance' to dataveillance and data mining. Who the individual 'is', where s/he is going, with whom s/he is associating or what s/he is doing all become the concern of the watchful eye [355].

In Internet Encyclopedia of Philosophy[1], Kevin Macnish writes:

"*Surveillance involves paying close and sustained attention to another person. It is distinct from casual yet focused people watching, such as might occur at a pavement cafe, to the extent that*

[1]Source: https://iep.utm.edu/surv-eth/

TABLE 2.1: Summary of video surveillance systems evolution

First-generation	*Technologies*	Based on analogue technologies
	Advantages	Very reliable Easy to use
	Problems	Data quality degradation Very high bandwidth requirement Storage challenge
	Current Research	Digital video recording, and video compression
Second generation	*Technologies*	A hybrid system based on analogue and digital technologies
	Advantages	Increase the efficiency of the video surveillance system Increase the quality of the stored video The ability to control the system remotely
	Problems	Inflexible system Higher maintenance cost Robust detection and tracking algorithms are required
	Current Research	Real-time computer vision algorithms Automatic learning of patterns of behaviours
Third generation	*Technologies*	Based on digital technologies
	Advantages	Ensures better reliability More efficient for large scale systems More accurate information Reduces the degree of deterioration in data quality Less bandwidth requirement
	Problems	More expensive for small surveillance systems Distribution of information (integration and communication) Design methodology Multi-sensor platforms
	Current Research	Distributed versus centralized intelligence Probabilistic reasoning framework Multi-camera surveillance techniques

it is sustained over time. Furthermore, the design is not to pay attention to just anyone, but to pay attention to some entity (a person or group) in particular and for a particular reason. Nor does surveillance have to involve watching. It may also involve listening, as when a telephone conversation is bugged, or even smelling, as in the case of dogs trained to discover drugs, or hardware which is able to discover explosives at a distance". (Kevin Macnish, [413])

In our context, we can define surveillance as monitoring behaviours or events in our environment or ourselves for information gathering, organisation, management, or direction using CCTV systems to determine a potential attack or collect data and use it where deemed necessary.

2.3.1.2 Counter-surveillance

In the short term, counter-surveillance is any action, which aims to strengthen private security to prevent or at least make it difficult. According to the International Protection organisation, counter-surveillance is *"the activity carried out to counter-surveillance or intelligence"*. Generally, counter-observation is a strategy that uses many forms of monitoring to alter the balance of power between the supervised and the watcher.

Torin Monahan in [287] defines counter-surveillance as the intentional or strategic use or interruption of surveillance technology to challenge institutional control's asymmetry. These activities can involve the disabling or destruction of surveillance cameras, the mapping, and dissemination of minimum surveillance paths over the Internet, the use of video cameras to track sanctioned surveillance systems and their employees, or the organisation of public theatre performances to draw attention to the prevalence of surveillance in society.

Several techniques and methods have been developed to achieve this goal. Some examples include creating new security protocols for exchanging information, advances in encryption, and enhanced security of operating systems and database management systems. However, despite all the community's efforts, the security of privacy remains a significant challenge that all stakeholders must confront.

2.3.1.3 Video surveillance

The term video surveillance refers to any software and hardware equipment that enables a government, organisation, individual, or group of individuals to set up a system that performs for surveillance purposes in public and private places as a valuable tool in the context of crime prevention and detection. According to Jordi et al. [13], video surveillance refers to a generic system that describes different agents' actions and interactions within complex scenes in real-time.

In our context, video surveillance consists of remotely monitoring public or private places, using video cameras that transmit videos or images to the surveillance station via cables or telephone links forming a closed circuit (CCTV). Video surveillance guides a range of security activities such as deterrence, observation, and intelligent collection, evaluating a possible and actual incident.

2.3.1.4 IP video surveillance or network video surveillance

IP video surveillance or IP CCTV refers to a security system that provides a user with the ability to monitor and record video/images over an Internet Protocol (IP)-based computer network, such as wireless networks, local area networks, or simply via the Internet. In addition to providing remote access, the IP CCTV allows real-time or non-real-time streaming from any device equipped with software capable of receiving video streams, such as a PDA or a simple tablet.

Unlike analogue CCTV systems, IP CCTV systems introduce many innovative features that enhance the monitoring and management of live and recorded video data, making them ideal for security surveillance applications. They include remote accessibility, high image quality, comfortable and future-proof integration, scalability, flexibility, cost-effectiveness, event management, and intelligent video.

2.3.1.5 Intelligent/Smart video surveillance

Intelligent video surveillance is introduced to ensure constant monitoring and remedy the issue of not missing a crucial incident. Intelligent video surveillance or video analytics is a technology that makes it possible, through video analytics technology, to identify specific events in video

sequences automatically. It converts video into data that will be distributed or archived to enable the video surveillance system to act accordingly. This could include activating a mobile camera to collect more reliable data from the scene, or merely transmitting an alert to the security staff so that they can decide on the required intervention to be taken.

The IBM research group [415] has identified an intelligent video surveillance system as a system capable of recording and processing video images to enhance the protection system. It is fitted with an embedded system that can perform several video processing algorithms. Intelligent video surveillance systems can perform different video analysis forms, including motion detection, object detection, tracking, automated search options, video playback, video storage index, etc.

According to Xiaogang Wang [441], intelligent video surveillance refers to the efficient extraction of information from the video data by the automatic detection, tracking, and recognition of objects of interest and the analysis and understanding of their activities and related events.

2.3.2 System overview and description of components

In this section, we briefly present the different hardware and software components of video surveillance systems. A more detailed description of the infrastructure of video surveillance systems and the principles guiding the choice of materials can be found in several books and reference guides, including the following [160, 172, 309].

As shown in Figure 2.1, standard video surveillance systems generally consist of the following main components: acquisition equipment, compression, transmission, processing or video management system, storage, and archiving.

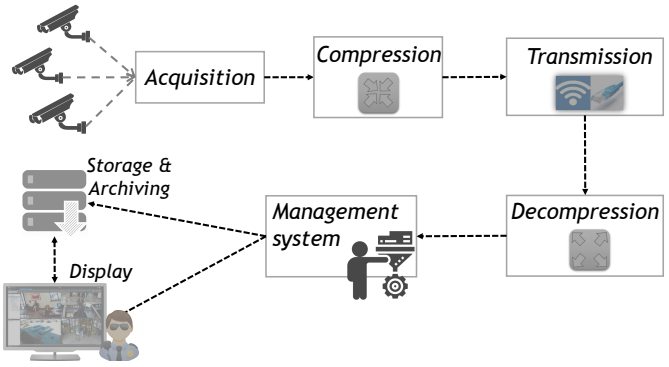


FIGURE 2.1: Main components of intelligent video surveillance

2.3.2.1 Acquisition equipment

This refers to the different camera models used to acquire the videos/images in the video surveillance system. Several models and forms of surveillance cameras in the real world are structured into two main categories: analogue and digital. This diversity of camera forms is due, on the one hand, to the manufacturers who strive to use good design in the world, and on the other hand, to the needs and conditions foreseen for the use of these cameras.

Before the appearance of IP cameras, analogue cameras were used. Systems based on this category use tapes to record the data acquired by these cameras. A digital video recorder (DVR)

converts analogue video to digital, replaces tapes during the second generation. Over the last generation, surveillance systems use IP (Internet Protocol) compatible cameras to transmit digital data over an Ethernet or wireless network. As discussed in section 2.2, the third generation is fully digital and is generally based on IP/network cameras. However, the new intelligent video surveillance systems use IP/network cameras alongside analogue cameras as a hybrid camera system to benefit from their advantages.

2.3.2.2 Compression

The data traffic generated by video surveillance systems would be extremely large and could overwhelm most networks, significantly increasing storage needs and costs. To overcome this problem, the use of compression techniques is essential. Compression techniques involve eliminating irrelevant portions of images or eliminating redundant portions of the media stream. However, in compression, the data loss rate must be taken into account, where a compromise must be made between file size and image quality.

In general, compression is the reduction of data transmitted between system devices. In analogue video surveillance systems, this compression occurs in DVRs. This compression occurs at cameras themselves in an IP/network video surveillance system, as shown in Figure 2.2.

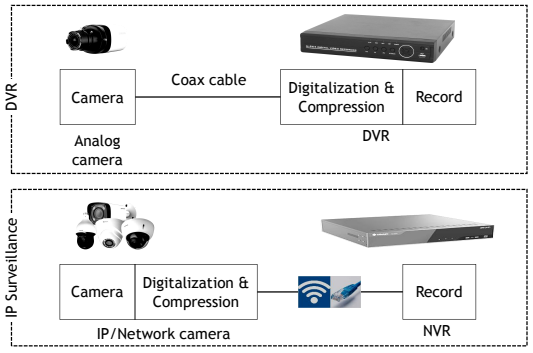


FIGURE 2.2: Compression level in the analogue and digital surveillance system

2.3.2.3 Transmission

For the recording, processing, and displaying video recorded by the camera, the latter transmits over the network from the camera to the surveillance unit. The transmission of this data is carried out by one of the following transmission techniques:

- ***Wired transmission***: The KX6 coaxial cable is the most widely used for the wired transmission of analogue video. It consists of a power cable and a video cable for transmitting images and sounds with a maximum distance of 250 meters.

  All analogue video transmission media can add an encoding/decoding module to the wire transmission of digital video. The latter allows converting analogue video to digital video and vice versa. For IP video surveillance systems, it is possible to use the Ethernet network, which offers a throughput of up to 1 Gbps and supports a considerable distance in the presence of switches and repeaters to regenerate the signal once weakened.

- ***Wireless transmission***: In this transmission mode, whether for digital or analogue video, wireless transmission requires a transmitter and a receiver of the signal. This type of transmission is proper when the use of cables is not possible or very expensive. Among the wireless technologies used in IP video surveillance networks are Wi-Fi and WiMAX, which are the most widely used and best-known. WiMAX technology is preferred over Wi-Fi for long distances because of its wider coverage, higher transfer rate, higher transmission volume, and better quality of service.

The wired transmission provides greater bandwidth and reliability at a reduced cost than wireless transmission, whereas wireless transmission is better when a large area is under surveillance or where cables cannot be accessed in the area under surveillance.

2.3.2.4 Video management system

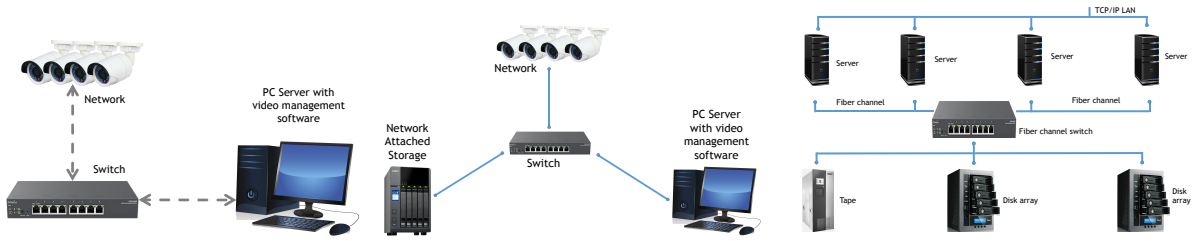
Video surveillance image processing is carried out at the processing unit, called the server, or at a smart camera. The processing process includes video streaming, viewing, recording, analysing, and searching for recorded footage. The four main categories of video management systems used in video surveillance systems are as follows:

- ***Digital Video Recorder (DVR)***: This device appeared on the market in 1999. A DVR is equipped with an internal hard drive for digital video recording and integrated video processing software. It just takes video from analogue cameras and digitises the video. The DVR is gradually disappearing due to the emergence of IP video cameras.
- ***Hybrid Digital Video Recording (HDVR)***: It is similar to the DVR, except that it can accept signals from analogue cameras and convert them to digital and the possibility of supporting IP cameras.
- ***Network Video Recorder (NVR)***: It is designed for an IP network architecture, and it only supports IP cameras. However, it requires an encoder to support analogue cameras.
- ***Super Digital Video Recorder (SDVR)***: In simple terms, Super DVR is a DVR with higher specifications due to the combination of DVR / HDVR / NVR functions.

2.3.2.5 Storage and archiving

In a typical surveillance system, servers and storage are essential components for monitoring, recording, managing and, archiving video. When deploying an efficient storage system, the number of cameras deployed, recording duration, recording criteria, and other factors must be considered. Video surveillance storage systems benefit from integrating standard server and storage components from the computer world. These can range from a low-cost desktop computer to a server system connected to a storage area network (SAN) with petabytes of storage that can cost millions of dollars. There are a few basic architecture types for storage systems, all with differences in complexity, performance, cost, redundancy, and scalability. The most common are described in the following:

- ***Direct Attached Storage (DAS)***: DAS is storage directly connected to the corresponding computer or server. In the context of video surveillance systems, we can distinguish two types:



(A) single-server storage (B) network-attached storage (C) Storage area network

FIGURE 2.3: The basic architectures of storage systems

- ❖ *Edge storage*: On-board or edge video storage means storage in the camera itself, i.e., temporary storage due to their low storage capacity. This type of storage can be a good choice for distributed video surveillance systems.
- ❖ *Single-server storage*: Storage on a single server is a possible solution for video storage in a small surveillance network (see Figure 2.3a). The hard disks are located on the same server that manages the video management software. These hard disks can contain terabytes or more space with long retention times.

❑ ***Network-attached storage (NAS)***: NAS is a server for storing computer data connected to a computer network that provides access to data to all hosts/clients on the network (see Figure 2.3b). This type of storage's advantages includes being very easy to install and manage, offering a cost-effective solution for storage needs, and suitable for large video surveillance systems, not forgetting its scalability, flexibility, and redundancy.

❑ ***Storage area network (SAN)***: Based on the same NAS-proven sharing principle, SAN is another particular high-speed network for storage devices. It is connected to one or more servers, as shown in Figure 2.3c. This topology allows users to access any of the storage devices on the SAN via the servers.

2.3.2.6 Display

In CCTV systems, the videos captured by surveillance cameras are eventually viewed for human beings on large screens to display some surveillance videos, which could also switch between several cameras and are generally used for previous investigations.

2.3.3 Analytics in video surveillance systems

Intelligent computer vision algorithms have been developed for automatic scene analysis to process all video surveillance data streams. From here, the concept of an intelligent video surveillance system (IVSS) emerged. IVSS offers systems for detecting and tracking objects and automatically reporting suspicious events without human intervention. The intelligent analysis methods and techniques used in IVSS belong to a recent technology called video analytics. This latter offers solutions to process the recorded video sequence to retain security-related data. It also improves search capabilities in archived videos.

Video analysis includes different levels of analysis and processing. Hierarchically, they are executed from the pixel level (low level) to the behavioural level (high level). These functions

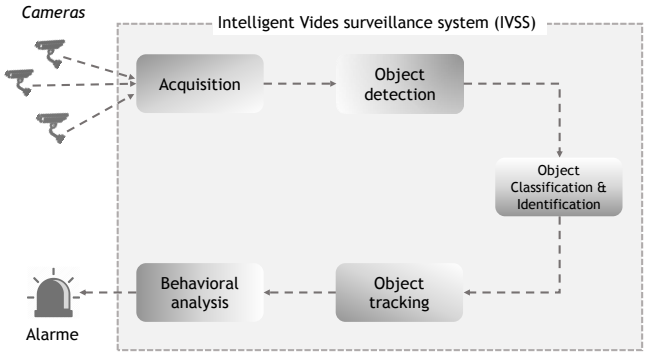


FIGURE 2.4: Video analytics tasks performed in intelligent video surveillance systems

can be restricted in the following tasks: detection, tracking, classification, and identification, as well as behavioural analysis of moving objects. The diagram in 2.4 shows the scheduling of these tasks.

2.3.3.1 Acquisition

The acquisition process starts with the cameras. In non-real-time systems, the cameras capture the images/frames and send them to the processing unit. At the processing unit, the received frames are stored in temporary memories, after which the processing system takes these frames from the tempo memory for further processing. The cameras do not capture any frames for real-time systems until the processing unit finishes processing the previous frames.

2.3.3.2 Object detection

Object detection is used to detect any specific activity in the surveillance area, such as the movement, appearance, or disappearance of objects. Detection is a difficult task; it generally depends on the scene's nature and the corresponding application.

In the literature, many motion detection techniques are based on change detection. The three main conventional approaches to object detection are temporal differencing, background subtraction, and optical flow.

- ***Temporal differencing***: This approach is based on a temporal differentiation of pixels between consecutive frames in a video stream. It extracts the moving regions by analysing the temporal variation of pixel light intensity. It is swift and adapts to dynamic environments. Mathematically, temporal differentiation consists of calculating the absolute difference Δ_t between n consecutive frames, as shown in equation 1.

$$\Delta_t(x,y) = |I_t(x,y) - I_{t-n}(x,y)| \tag{2.1}$$

  With, $I_t(x,y)$ is the luminous intensity of the pixel of coordinates (x,y) of the t^{th} image and $n \in [1,5]$. A fixed threshold τ allows distinguishing between the pixels belonging to the foreground and those belonging to the background. The image of the foreground areas $F(x,y)$ is extracted by thresholding (see equation 2).

$$F_t(x,y) = \begin{cases} 1 & \text{if } \Delta_t(x,y) \geq \tau \\ 0 & \text{else} \end{cases} \tag{2.2}$$

Temporal differentiation adapts to changes in the environment over time. On the other hand, it tends to neglect certain variations related to the movement of objects in the scene, especially if they move slowly. Indeed, it often produces holes in the detected objects. This technique, therefore, requires a smoothing treatment, with morphological operators and filtering of holes and sizes that are too small. According to the movement pattern, some techniques allow mapping areas with activity levels to focus only on large movements and eliminate occasional movements.

- ***Background subtraction***: The second technique is based on a background's subtraction or a reference model from the current image. It is widely used in systems based on fixed cameras [138–140, 301, 302]. The background subtraction consists of two steps: background modelling and motion segmentation. Background modelling involves differentiating between background and foreground objects or structures. The foreground is the part of the image that is closest to and in front of the viewer, while the background is the set of fixed objects along with the life of the system. The modelling of the background creates, as a result, a reference image representing the background. Motion segmentation involves the extraction of moving objects from a reference background.

  This method, in its simple version, consists of making the difference between the intensities of the pixels $I(x,y,t)$ of an image I took at time t, compared to the intensities of the pixels $B(x,y)$ of a reference model of background B. It detects pixels (x,y) that have undergone a significant change in intensity. These pixels form the moving objects. This is modelled by:

$$\begin{cases} pixel(x,y) \in motion & \text{if } |I(x,y,t) - B(x,y)| > \tau \\ pixel(x,y) \in background & \text{else} \end{cases} \tag{2.3}$$

  Background subtraction has better performance in extracting information about objects, but it is sensitive to dynamic changes or partial movements in the environment and changes in climate or light.

- ***Optical flow***: The optical flux is a dense field of displacement vectors that define the transposition of each pixel in a region to describe the temporal directional rate of pixels in two successive frames. A two-dimensional velocity vector carrying information about the direction and speed of motion is assigned to each pixel in the frame. It is calculated using the luminosity constraint, which assumes that the pixels' brightness, corresponding to consecutive frames, is constant.

  To have a simpler and faster calculation, the real world in three dimensions is transferred in two dimensions. Then the image is described using a dynamic luminosity function $I(x,y,t)$ dependent on pixel coordinates and time. Assuming that in the neighbourhood of a moving pixel, the change in intensity of brightness does not occur along the field of motion, the following expression is deduced:

$$I(x,y,t) = I(x+\delta_x, y+\delta_y, t+\delta_t) \tag{2.4}$$

  Applying the Taylor series to the right side of equation 4, we obtain:

$$I(x+\delta_x, y+\delta_y, t+\delta_t) = I(x,y,t) + \frac{\delta_I}{\delta_x}\delta_x + \frac{\delta_I}{\delta_y}\delta_y + \frac{\delta_I}{\delta_t}\delta_t + H.O.T^{2} \tag{2.5}$$

From equation 4 and 5, neglecting the higher order terms ($H.O.T$) and after modifications, we obtain:

$$I_x \cdot v_x + I_y \cdot v_y = -I_t \tag{2.6}$$

Which is translated in the formal representation of the vector by:

$$\nabla I \cdot \vec{v} = -I_t \tag{2.7}$$

Where ∇ is the spatial gradient of the brightness intensity, $\vec{v}$ is the optical flow (velocity vector) of the pixel, and I_t is the time derivative of the brightness intensity.

Optical flow is used to detect moving objects independently in the presence of camera movement. However, most of its methods require complex calculations that are difficult to perform in real-time. In addition, optical flow is sensitive to image noise [68].

2.3.3.3 Object classification and identification

Objects detected in the previous phase will usually be classified into different classes or categories: human, vehicle, animal, etc. This classification can be done before tracking to keep only the trajectories of relevant objects for surveillance purposes. To implement the latter, the system goes through the following steps:

- ***Object segmentation***:

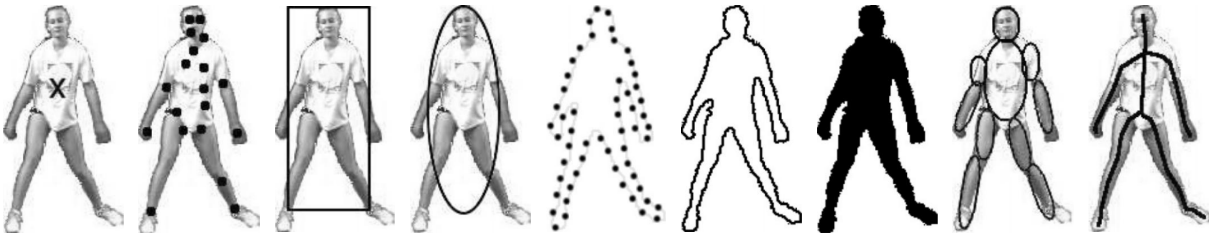
  After detecting moving objects, the foreground pixels are grouped according to the pixel type. This operation is called segmentation, and its purpose is to resemble the pixels to each other according to specific predefined criteria (such as colour, intensity, texture, etc.) to build so-called segments or regions. As a result, adjacent regions are substantially defeated concerning the same criteria. These regions are then grouped according to the degree of similarity into homogeneous regions called blobs.

- ***Object representation***:

  The blobs created by the previous step are very useful in the classification and tracking steps. However, blobs take up a large area of pixels and memory to store and display them due to their formation, so compacting the blobs to represent the objects is necessary. In the literature, several forms are commonly used to represent the detected object:

  - *Points*: the object can be represented by a point such as the centre of gravity, the centre of its enclosing box, or represented by several points (Figure 2.5a, 2.5b). This approach is most commonly used because of its simplicity of processing with complex algorithms.
  - *Primitive geometric shapes*: this representation consists of grouping the totality of the detected object's points in the minimum possible form. In general, the shapes of the object are represented by rectangles, ellipses (Figure 2.5c, 2.5d), or other geometric shapes.

[2] H.O.T: Higher Order Terms



(A) (B) (C) (D) (E) (F) (G) (H) (I)

FIGURE 2.5: Object representations (from [488])

- ❖ *Object silhouette and contour*: this approach is very suitable for tracking non-rigid and complex shapes. It uses contours to define the boundaries or ends of the object (Figure 2.5e, 2.5f), and all that exists inside the contours are called silhouettes (Figure 2.5g).
- ❖ *Articulated shape models*: articulated objects are composed of body parts that are held together with joints. The simplest example of this representation is the human body. To represent an articulated object, the constituent parts can be modelled using cylinders or ellipses, as shown in Figure 2.5h.
- ❖ *Skeletal models*: in this representation, an object's skeleton can be extracted to model articulated and rigid objects (see Figure 2.5i). The skeleton can be defined as a set of articulations that describes the dependencies and defines the constraints between the parts' representations.

❑ ***Object classification***:

Object classification consists of distinguishing the different objects present in the image into predefined classes such as human, vehicle, animal, disorder, etc. The four most commonly used approaches to object classification are:

- ❖ *Shape-based classification*: shape-based classification is purely concerned with the geometry of the object. Objects can be classified depending on the extracted regions' geometry, such as enclosing boxes and external contours.
- ❖ *Motion-based classification*: the motion-based approach provides a robust classification method. It does not require predefined shape models, but it has difficulty identifying a non-moving object. Although motion-based classification has moderate accuracy, it is inexpensive to compute.
- ❖ *Texture-based classification*: assigning an image to a known texture class is an essential objective of texture-based classification. With several classifiers, the main task is the extraction of the pertinent characteristics of the textured image. These methods give a better accuracy, but with an additional computing time.
- ❖ *Colour-based classification*: unlike other approaches, colour is relatively constant during changes in perspective and is easy to acquire. The representation of colour characteristics is the most effective way to reveal the similarity of colour images, but the accuracy and computational time are high for colour-based classification.

❑ ***Object identification***:

In the IVSS, to track the detected objects, their identities must be known. After identifying the class to which these objects belong in the classification phase, the system moves on to the identification stage through face recognition for people, license plate reading for cars, etc.

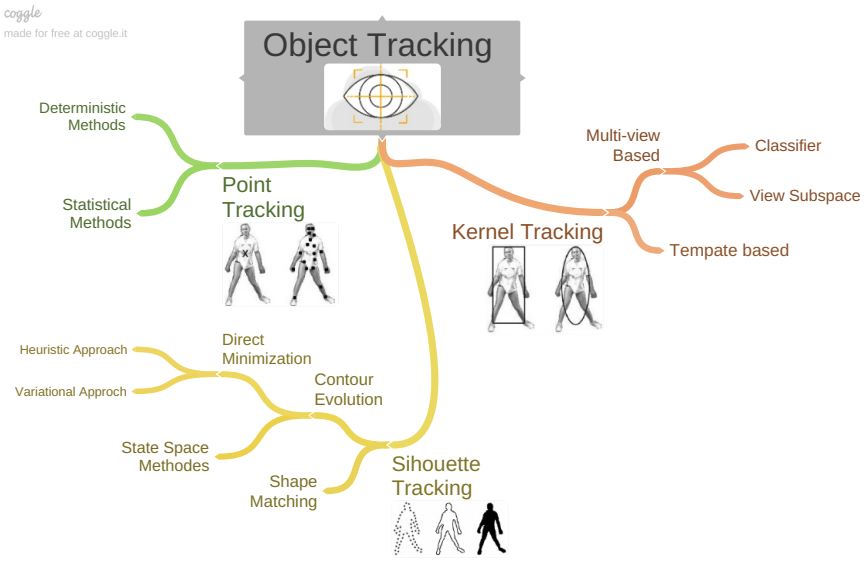


FIGURE 2.6: Classification of tracking methods (from [488])

2.3.3.4 Object tracking

Object tracking consists of tracing their trajectory in a well-defined period by locating their position in the frame at each instant, i.e., the regions occupied by the object at each instant along the video stream. This operation requires finding each object from one frame to the next. Tracking can be done in 2D, from a single camera, or in 3D, combining two views with a known geometric relationship.

This step's general principle is to establish a correspondence between two instances of the same object in the next frame. Many tracking techniques are based on mathematical methods that predict an object's position based on its movement history observed in previous frames. Tracking moving objects is a complicated operation due to many factors such as the complexity of the shape, the non-rigid nature, the movement of objects, partial or complete occlusions, changes in scene lighting, etc.

Based on Yilmaz et al. [488] classification, object tracking approaches can be classified into three categories, as shown in Figure 2.6. The criterion used for this classification is based on the representation of the object discussed above.

- ***Point Tracking***: is one or more points represent a method where objects are detected, and the association of these points is based on the previous state of the object. This approach is complicated and sometimes impossible to perform a correspondence between the points, especially in the presence of occlusions, wrong detections, the presence of a new object, and the exit of this object from the scene.
- ***Kernel Tracking***: is based on the kernel measure of the emerging region of a moving object between frames. The term "kernel" in our context refers to the shape and appearance of the object. The object's motion can be a parametric motion such as translation, conformal, affine, etc., or the dense flow field calculated in the next frame. Technically, it is a matter of measuring the object's movement using geometric shapes such as rectangles and ellipses. This technique's major disadvantage is the inability to differentiate between parts of the object or backgrounds that overlap when the object is moving.

- ***Skeletal models***: in most cases, objects do not have simple geometric contours, such as a human body with complex geometric shapes. The silhouette-based object tracking method aims to find the object in context from a region in each frame using the object models generated by previous images. The information can be in the form of appearance density or shape model, usually represented by contour maps. Silhouette-based tracking allows objects of this type to be tracked correctly, as it can support accurate shape descriptions for objects.

2.3.3.5 Behavioral analysis

Behavioural analysis is the highest-level task of intelligent video surveillance systems to interpret an object's behaviours, group of objects, or crowds in a scene and their interactions. In the video surveillance context, behaviours are defined as observable actions of objects. This task requires a semantic analysis, sometimes complex, depending on the application's context and the detected event.

The most widely used techniques for modelling normal behaviour and detecting deviant behaviour are hidden, Markov models, neural networks, and Bayesian networks. These techniques generate an alarm based on the static divergence from the inferred model of the scene. There are also predefined event detection methods. They are based on a system of rules. For example, generate an alarm if an object larger than a threshold remains stationary for more than a specific time in a given region.

2.3.4 Video surveillance system classifications

To date, a wide variety of surveillance systems are available. These systems may be classified according to different criteria, such as automation level [161], network architecture [161], application area [86], application-level [49], outdoor and indoor [344], and computing paradigms. Figure 2.7 illustrates the classification and its criteria.

2.3.4.1 Classification based on the automation level

Kim in [216] classify video surveillance systems into three categories according to the level of autonomy of the system, namely: manual, semi-autonomous, and fully autonomous.

- ***Manual video surveillance systems***: This type of system is the most used to date in reality. Manual systems are fully managed by qualified human operators who monitor and analyse video recordings displayed on multiple screens. This operation requires additional human resources and constant monitoring. As a result, fatigue and the resulting distraction allows for common human errors.
- ***Semi-autonomous video surveillance systems***: They are hybrid systems that combine video processing and human intervention. In these systems, some tasks are performed automatically by the system, and human operators act on others manually. The simplest example of this system is a system that only records video containing the significant movement and analyses it by surveillance personnel.

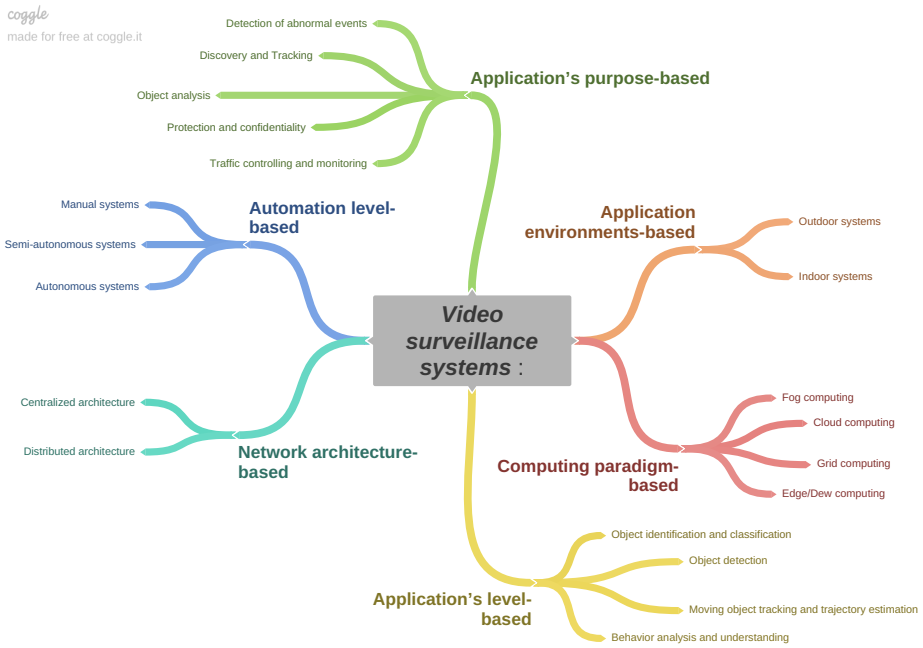


FIGURE 2.7: Map of video surveillance system classifications

- ***Autonomous video surveillance systems***: Fully autonomous systems are systems that do not require the intervention of human operators. The system itself performs all tasks in real-time, such as detection, motion tracking, and decision-making.

2.3.4.2 Classification based on network architecture

Gouaillier in [161], classify video surveillance systems into two main categories according to the system architecture.

- ***Centralized architecture***: In this type of architecture, all the cameras deployed in a space such as a company or a city send the video sequences to a BS. All video processing is done at this station. Concerning the connection of the cameras with the control station, they are either linked by an IP network or directly connected to the station, as discussed in section 2.3.2.3.

  Among the drawbacks of this type of architecture, we mention:

  - ✗ This system requires significant storage resources and processing capacity to manage the video streams from a large number of cameras.
  - ✗ The transmission of camera streams to a single station in a centralised network requires bandwidth, especially in networks with many cameras. It is also causing bottlenecks that reduced the systems' performance, especially in terms of real-time processing.
  - ✗ All system resources are carried out in a single processing station. Because of this centralisation, the whole system stops if this station fails.

- ***Distributed architecture***: The distributed architecture of the video surveillance system is based on the principle of load sharing, where video processing and storage can be

shared or distributed across different system nodes. This architecture aims to use all the resources available in the system that are not used in the centralised architecture to provide additional resources.

The advantages of this architecture, compared to the centralised architecture, are summarised in the following points:

- ✓ Supports real-time recording without delay and eliminates the bottleneck problem.
- ✓ Reduces bandwidth and network congestion.
- ✓ Ensures system reliability and facilitates the expansion of the camera network.

2.3.4.3 Classification based on the application's purpose

According to Chamasemani [86], video surveillance systems can also be classified according to their field of application or objective. With this criterion, video surveillance systems can be divided into five categories.

- ❏ ***Protection and confidentiality***: This type of system was developed for the protection of people in both public and private places. Governments commonly use these systems for territorial security [194], and the security of public places (airports, shopping malls, etc.) [85, 306], or by citizens for home surveillance [347].
- ❏ ***Object analysis***: This system's objective is to analyse objects for classification, recognition, and analysis of their behaviour. This category includes facial recognition systems [10], license plate recognition systems [456], moving vehicle classification systems [462], etc.
- ❏ ***Discovery and Tracking***: This category includes any research focused on identifying, tracking, and monitoring activities of interest such as pedestrian tracking, gait-based analysis and identification, autonomous navigation, monitoring of a complex environment, etc.
- ❏ ***Traffic controlling and monitoring***: Automatic traffic monitoring plays a vital role in our lives for road traffic control. These systems address problems such as estimation of vehicle flow rate [515], road network protection, automated vehicle speed control, highway traffic density [373], and marine traffic [326].
- ❏ ***Detection of abnormal events***: These systems aim at monitoring environments, detecting abnormal events and alerting in special cases, such as: fire detection [147, 177], accident detection [137, 262], crime detection [386], etc.

2.3.4.4 Classification based on the application's level

Ben Hamida in [49] discovered a new classification of video surveillance systems based on application-level or processing level. As seen in section 2, video analytics in IVSS starts from the low level or pixel level for object detection to the highest level for analysing and interpreting the object's behaviour. Based on this processing hierarchy, video surveillance systems can be grouped into four categories, as follows:

- ***Object detection***: This category of systems is based on object detection techniques only, without proceeding to higher-level functions. These applications are used for simple tasks such as monitoring bus and train station terminations [179], alerting to activity on the site [16, 276], and counting incoming and/or outgoing people or vehicles to estimate traffic congestion and the flow of vehicles on a particular road [354, 422].
- ***Moving object tracking and trajectory estimation***: This type of system is designed to track and estimate the trajectory of objects appearing in scenes. As examples of the application, traffic monitoring systems, elderly monitoring systems, and systems for monitoring workers' activities in high-risk locations such as chemical and nuclear plants, and systems for tracking abandoned or stolen objects.
- ***Object identification and classification***: In most video surveillance systems, the objects detected or tracked are classified into different categories. To do this, the system must recognize the nature of the detected object based on the attributes of its shape or the properties of its movement or other features. Identifying persons and vehicles authorized [1] by facial recognition and license plate recognition are the two most common systems in this category.
- ***Behaviour analysis and understanding***: This system is based on the understanding and interpretation of objects' behaviour through the combination of information from lower-level functions and high-level semantic interpretation. Among the systems developed in this category, particularly tracking suspicious objects, detecting missing persons, and surveillance older adults.

2.3.4.5 Classification based on the application environments

Another classification criterion that can be added to the other previous criteria mentioned above is the application environment. Based on this, we can divide the systems into two categories: outdoor and indoor video surveillance systems.

- ***Outdoor video surveillance system***: This category includes all video surveillance systems installed in places or locations away from building boundaries. These systems are very complex and generally deployed over large monitoring areas, requiring significant resources. This category includes road monitoring systems, forest fire protection systems, and systems for detecting, tracking moving objects in public places, and border intrusion detection systems. These systems offer a high degree of extensibility and adaptability to weather conditions and changing lighting conditions.
- ***Indoor video surveillance system***: This category includes applications for interior spaces such as homes, hospitals, museums, etc. Systems of this category are less complex systems, deploying over limited areas and requiring fewer resources than the previous system. Some examples of this kind of system include patient monitoring systems, home intrusion detection systems, crowd and queue control systems in airports.

2.3.4.6 Contribution to classification based on the computing paradigm

To date, several computing paradigms have emerged in the context of the IoT (sections 1.3 and 1.4 of chapter 1). These paradigms are used in various application domains to improve

service quality performance in computing power, storage capacity, and connectivity efficiency. Video surveillance systems are one of the application areas that have greatly improved due to these paradigms. Depending on these systems' needs and requirements, such as storage capacity, connectivity reliability, and real-time execution, it adapts the most appropriate paradigm. Depending on the paradigm employed, we can classify video surveillance systems into four categories: video surveillance systems based on (*i*) the cloud, (*ii*) the cloudlet, (*iii*) the fog, and (*iv*) the edge/dew computing.

- ***Video surveillance system based on cloud computing***: This category generally includes video surveillance systems that require a large storage capacity to store massive multimedia data, considering aspects of security, reliability, and fault tolerance. This category also includes systems that require higher processing capacity to run complex video analytics algorithms. Most systems in this category do not require real-time processing as a result of the higher time latency problem of cloud computing (see section 1.3 of chapter 1).
- ***Video surveillance system based on cloudlet computing***: This system usually requires fast communication and execution in addition to storage and processing. The advantage of a cloudlet or a small data centre over these systems is reducing system load and reducing latency. Therefore, these systems are near-real-time systems, i.e., systems that do not require a large amount of bandwidth to transfer their multimedia data over the network.
- ***Video surveillance system based on fog computing***: Video surveillance systems based on the fog computing paradigm are large-scale and distributed systems that support real-time monitoring and cooperation with other heterogeneous systems. Most systems in this category share or distribute system tasks or processing between the fog and the different paradigms if it supports them. Time-sensitive data is processed and stored at the fog computing level, and the other tasks are shared in other paradigms.
- ***Video surveillance system based on edge computing***: Systems in this category are distributed dynamic systems based mainly on intelligent or embedded cameras with processing, storage, and communication capabilities. Despite these meagre resources, it plays an essential role in real-time systems that require high bandwidth and a network supporting a large load and security and confidentiality.

2.3.5 Requirements for next generation video surveillance

With this large number of cameras and the enormous amount of data generated, camera management, transmission, storage, and analysis of data could pose many significant challenges to existing systems, which are summarized in the following points:

- ***Network***: The ultra-big multimedia data generated by the surveillance cameras is transmitted and aggregated on a central server or processing unit. Counting all system functionality on a single server may cause bottleneck congestion and latency problem that affects the system's reliability. In this case, conventional architecture is unable to solve these problems. Emerging wireless communications technology deployments such as 5G and 6G may offer a promising solution for these challenges.

- ***Architecture***: With the rapid development of camera networks in terms of size and complexity, the requirements for robustness, reliability, scalability, transferability, and self-adaptation are higher. With the new computing paradigms that have emerged in the IoT arena, distributed intelligent video surveillance systems offer a promising solution to address these requirements and challenges. The latter distributes the system load over the system's different layers through the different emerging paradigms. For the latter solution, several issues need to be considered for video surveillance systems:
  - Is it possible to implement the system at multiple levels?
  - How to partition the system load between these levels?
  - To reduce the use of network bandwidth, overall cost, and efficiency, how to select the partition and the steps to be performed?
- ***Optimization***: There is a strong need to increase the amount of storage available to meet the requirements, and it is costly. For that, it is necessary to reduce the amount of video data stored by saving only permanent information and avoid information being expurgated and useless (for example, the same person being followed by serval cameras at the same time or videos that do not contain motion). Unfortunately, after generating this useless data, it is too late to filter it because we will spend more processing resources and energy after losing those resources and the storage and bandwidth to generate it for the first time. Therefore, a new strategy must be adopted to avoid generating these data types from the source (i.e., at the camera level).
- ***Real-Time***: The high complexity of intelligent video surveillance system tasks (detection, recognition, tracking, and analysis) on the server can lead to image or video processing delays, which means that the system is not suitable for real-time operation.
- ***Security***: Another important aspect is the security of the communication. Data may need to be sent over open networks for some visual surveillance systems, and there are critical issues in maintaining confidentiality and authentication.

2.4 Distributed multi-Camera coordination for IVSS

During the last few years, multi-camera surveillance systems have been the subject of particular attention in research and industry. Today, most commercial video surveillance systems rely on a conventional centralized architecture to perform surveillance tasks. To support the more complex and advanced video surveillance systems offered in recent years, companies must use distributed intelligent multi-camera coordination systems.

This emerging system is a real-time distributed onboard system that performs computer vision using multiple cameras. This new approach was born thanks to the fusion of concurrent developments in four major disciplines: computer vision, embedded sensors, computing paradigms, and networks. Processing images in a distributed smart camera network poses many problems and challenges. However, we believe that the issues addressed by these systems are more important to improve the efficiency and QoS of video surveillance systems. We agree with Bernhard and Wayne[350] that distributed smart cameras with a coordination mechanism are critical components for the next generation of computer vision systems and that smart cameras will become an enabling technology for many new applications. This section presents a review of the literature on distributed multi-camera coordination for IVSS.

2.4.1 Basic concepts definitions

2.4.1.1 Distributed systems

❑ ***Definition***

Distributed systems are systems consisting of different entities[3], physically distributed but interdependent. They work on different networked computers, but they have to exchange and share information to work together towards a common goal.

Distributed systems have many different aspects that are very difficult to understand by a single definition. One definition of distributed systems is given by Sander Tichelaar [416] in a study of open distributed systems and coordination. They say:

"*A distributed system is a collection of loosely coupled entities in a distributed environment, working together to achieve a common goal*". (Sander Tichelaar, [416])

The book "Distributed Systems for System Architects" by Paulo Verissimo and Luis Rodrigues [430] consists of another definition of distributed systems close to the previous one, which is as follows:

"*A distributed system is a system composed of several computers which communicate through a computer network, hosting processes that use a common set of distributed protocols to assist the coherent execution of distributed activities*". (Verissimo et al., [430])

Recently, Steen and Tanenbaum [427] suggest a new definition of distributed systems. This definition has another characteristic that was missing:

"*A distributed system is a collection of independent computers that appear to the system's users as a single computer*". (Steen & Tanenbaum, [427])

This definition highlights two important aspects:

- ❖ *"Autonomous machine"*: this point focuses on hardware resources, which can be of different types (heterogeneous) deployed in a work environment. Each processing node or entity is autonomous and interconnected by wired or wireless physical link media.
- ❖ *"A single coherent system"*: the use of the distributed resource set is managed by a system that provides a simplified view of the whole by hiding the distribution and heterogeneity of resources so that users view the system as a single computer.

❑ ***Motivation to use***

Among the reasons for deploying distributed systems are to share information and resources, increase reliability and performance, etc. [410].

- ✓ *Information exchange*: When several different nodes are connected, they will be able to communicate with each other to communicate or exchange data if necessary.
- ✓ *Resource sharing*: Consider a distributed system that links different systems or nodes with different capabilities. Clients can then use a resource from another node, such as a database.
- ✓ *Increased reliability*: If some nodes in a system may fail due to an overload of tasks in progress, the other nodes that are still functioning correctly can take over the tasks of the failed nodes. This will increase the reliability of the system.

[3] other common names are "nodes", "agents", "actors", "computers", "process", "sites", etc.

✓ *Increased performance*: If many jobs or tasks are partitioned into multiple independent, autonomous nodes (executed in parallel), the overall performance of a system can be better than if these tasks were performed sequentially.

❑ ***Communication modes in distributed systems***

The communication in distributed systems allows the entities to complete their knowledge and thus solve their assigned tasks. In the literature, two communication models are commonly used: shared memory and message passing.

❖ *Shared memory*: This model was introduced by Dijkstra in 1972 [124]. In this communication model, each entity e_i on a distributed system S communicates with neighbouring entities indirectly using shared memory locations, called registers. These registers allow each entity to read the status of all its neighbouring entities. More precisely, an entity e_i has access to two types of registers with each of its neighbour's e_j: a read register r_{ji} where e_i can only read the value of this register but not modify it, and a write register w_{ij} where e_i can read and change the value of this register. In this way, each entity e_i uses the write registers w_{ij} that it modifies to communicate the new values of its variables to its neighbouring processors e_j.

❖ *Message passing*: The message-passing or simply message-based communication model is the most widely used model in distributed systems. In this model, an entity e_i communicates with neighbouring entities e_j directly by sending and receiving messages through communication channels that can be unidirectional or bidirectional according to FIFO (First-In-First-Out) property.

In this model, the most common communication modes are the following two modes:

① *Point-to-point mode*: Each $e_i \in S$ can only communicate directly with its neighbours e_j.

② *Broadcast network mode*: Each $e_i \in S$ can communicate the same message simultaneously to several receivers $e_j \in S$. However, many variations, depending on whether or not multiple receptions in one entity create collisions and whether or not the collisions are detectable.

2.4.1.2 Distributed algorithms

A distributed algorithm A, on a distributed system S, is a set of instructions to be executed simultaneously on several entities $e_i \in S$. Each entity can calculate, communicate, and collaborate with other entities to accomplish a well-defined global task. The algorithm at the level of an isolated entity is called a local algorithm, and it usually corresponds to a classical sequential algorithm. The set of local algorithms together constitutes a distributed algorithm. If all local algorithms are identical, then the algorithm is said to be a uniform distributed algorithm.

Distributed algorithms are differentiated by the mode of communication used by the different entities of S (shared memory, message exchange), by the temporal model followed (synchronous, asynchronous), and by the nature of the implemented network (reliable, unreliable, anonymous,..., etc.). The properties cited previously are detailed in the following:

- ***Communication mode***: discussed in section 2.4.1.1.
- ***Synchronous communication***: is an instantaneous (real-time) communication between two or more entities. The entities must synchronize before communicating with each other.

- ***Asynchronous communication***: is a communication that takes place on a delayed basis without a global clock, i.e., spatial or temporal constraints are eliminated. A message sent by one entity arrives at the other entity at an arbitrary time.
- ***Reliable network***: is not only constantly available. A network is reliable if it offers sufficient capacity for lossless transmission or duplication of messages, whatever the circumstances.
- ***Anonymous network***: is defined as one where not all entities in the network have a unique identifier.

2.4.1.3 Distributed artificial intelligence

The Distributed Artificial Intelligence (DAI) appeared in the 1970s with the distribution of computing units and the parallelization of algorithms. DAI studies how a group of intelligent entities share resources and coordinate their activities so that the group's intelligence is superior to the intelligence set of the individuals that compose it.

DAI is an approach to solving complex problems like large-scale planning and decision-making problems. It is capable of managing large-scale computations and the spatial distribution of computing resources. These properties make it possible to solve problems requiring the processing of enormous data sets. It consists of autonomous, distributed processing entities. DAI entities can act independently, and partial solutions are integrated by communication between the nodes. Also, DAI systems are designed to be potentially adaptable to changes in the definition of the problem or the underlying data sets due to the scale and difficulty of redeployment.

2.4.1.4 Real-Time systems

Real-time computing or reactive computing is the computer science term for hardware and software systems subject to a "real-time constraint". Usually, real-time response times are of the order of milliseconds and sometimes microseconds. To build real-time systems, it is necessary to use design and implementation methodologies that guarantee the property of respecting time constraints. For example, a missed deadline can be catastrophic in this system, as in air traffic control systems or automotive systems.

The literature offers many definitions of real-time systems [77, 134, 400]. The functional definition proposed by J. Stankovic [400] is as follows:

"A real-time computer system is defined as a system whose correctness depends not only on the logical results of computations but also on the physical time at which the results are produced." (Stankovic, [400])

J-P provides the operational definition of a real-time system [134]. The latter is as follows:

"A real-time system is defined as any application implementing a computer system whose behaviour is conditioned by the state dynamic evolution of the process which is assigned to it. This computer system is then responsible for monitoring or controlling this process while respecting the application of temporal constraints." (J-P.Elloy, [134])

J-P.Elloy's definition clarifies the meaning of the term "real-time" by altering the relationships between computer systems and the external environment.

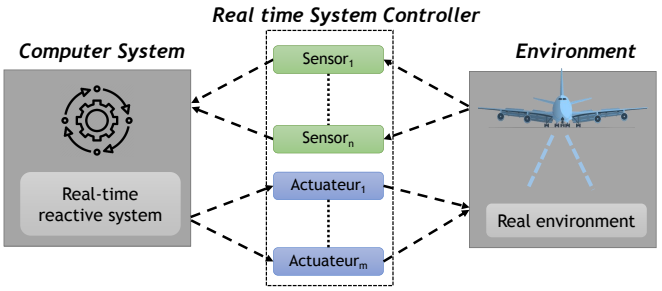


FIGURE 2.8: Schematic representation of a real-time system

The real-time systems are reactive [58, 175, 400]. The formal definition of a reactive system was given in [400]:

"A reactive system is a system that reacts continuously with its environment at a rate imposed by this environment. It receives inputs coming from the environment called stimuli via sensors. It reacts to all these stimuli by performing a certain number of operations and. It produces through actuators outputs that will be used by the environment. These outputs are called reactions or commands." (J. Stankovic, [400])

A real-time system usually needs to react to every stimulus coming from its environment. Moreover, the system's response depends not only on the input stimuli, but also on the system's state when the stimuli arrive. The interface between a real-time system and its environment consists of two types of peripherals. Sensors represent input devices used to collect a flow of information from the environment. Actuators are output devices that provide the environment with commands from the control system (see Figure 2.8).

2.4.1.5 Cooperative systems

❑ ***Definition***

A cooperative system comprises dynamic multi-entities that share information or tasks to decide or accomplish a common goal. Some cooperative control systems might include robots operating in a manufacturing cell, drones in search and rescue operations or surveillance missions, etc. [293]. The cooperative system's goal is to take another step towards an environment based on information, advice, and system deployment communication. The idea is to make the entities talk and coordinate with each other. This means that an entity should send and receive information from neighbouring entities through existing technologies and/or new technologies, depending on the requirements. The amount of information may indicate the level of cooperation exchanged between entities. Finally, cooperative systems may involve task sharing and consist of heterogeneous systems if they are composed of humans and machines [293].

According to the previous discussion, two main components cannot be ignored for the deployment of a cooperative system: communication and coordination, in addition to the other components listed in Figure 2.9.

❑ ***Why coordinate?***

Coordination between entities of the cooperative system is crucial due to four main factors:

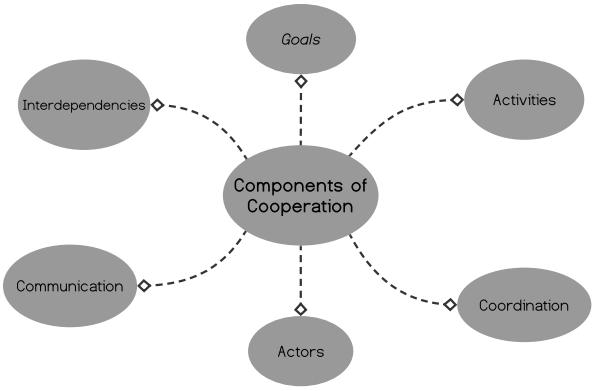


FIGURE 2.9: Components of cooperation

- ✓ *Incomplete knowledge* The entities of a cooperative system need information and knowledge that only other agents can provide to complete their global/partial vision of the system's state or for decision-making purposes.
- ✓ *Limited resources*: The local resources in each entity are limited. It is, therefore, necessary to share these resources between the other entities to optimize the actions to be carried out while trying to avoid possible conflicts.
- ✓ *Optimize costs*: The cooperation between entities also reduces costs by eliminating unnecessary and/or redundant activities.
- ✓ *Overcome dependence on objectives*: Allow agents with distinct goals but dependent on each other to meet those goals and perform their work, possibly taking advantage of that dependence.

2.4.2 Smart cameras and embedded computer vision

Intelligent/smart camera is usually used to refer to a camera with processing, storage, and communication capabilities (either in the same housing as shown in Figure 2.10 or close). Its first appearance dates back to the mid-80s. These cameras can automatically identify abnormal behaviour thanks to integrated intelligent algorithms and can extract information based on captured images or make an intelligent decision that will be applied in an automated process. At the beginning of its invention, there was a limitation of its capabilities in terms of sensitivity and processing power, but later, there were significant improvements in its capabilities. This type of camera is currently widely used in distributed architecture video surveillance systems thanks to their ability to communicate with heterogeneous systems such as other external devices, systems, or services.

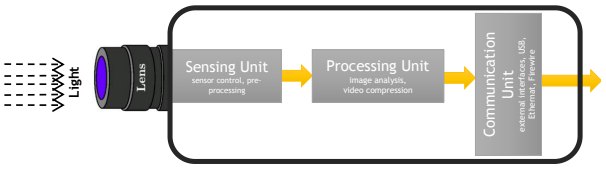


FIGURE 2.10: Smart camera architecture (from [350])

Figure 2.10 shows a generic smart camera architecture consisting of a module for sensing, processing, and communication. *(i)* The image sensor is the data source of a smart camera-processing pipeline, implemented in either CMOS[4] or CCD[5] technology. *(ii)* The Sensor module reads the raw image sensor data and performs some pre-processing in many cases, such as white balance and colour transformations. This module also controls important sensor parameters, such as capture rate, gain, or exposure, via a dedicated interface. *(iii)* The Processing module, which receives the collected images from the sensing unit, executes real-time image analysis and passes the abstracted data to the communication module. *(iv)* The communication module supports various external interfaces such as USB or Ethernet.

Smart cameras provide abstract data of the observed scene. Naturally, the abstraction provided depends on the architecture and application of the camera, and almost every smart camera provides a different output. Smart cameras execute various image processing algorithms, including motion detection, segmentation, tracking, object recognition, and so on. They usually deliver colours and geometric characteristics, segmented objects, or relatively high-level decisions such as wrong-way drivers or suspicious objects. Abstract results can be transferred either in the video stream or as a separate data stream. However, the embedded IT infrastructure of smart cameras is often exploited to perform high-level video compression and transfer only the compressed video stream.

Smart cameras have already been used in many applications. In the remainder of this section, we present a brief assessment of representative cases performed in different systems categories. This overview is borrowed from the article by Rinner, and Wolf [350].

- ❏ *Intelligent Video Surveillance Systems (IVSS)*: The fundamental task of the IVSS is to detect suspicious activity in the observed scene. This includes a complex analysis of images, starting from motion detection to segmentation, features extraction, and classification. To extend the coverage of the spatial sensor, the IVSS is a typical example of multi-camera systems. Most conventional IVSS, on the other hand, need little to no coordination among individual cameras.

- ❏ *Intelligent Transportation Systems (ITS)*: Infrastructure-based systems and vehicle-based systems are two types of vision systems in ITS. Infrastructure-based systems are typically large fixed multi-camera deployments. In this category, image analysis tasks are focused on traffic monitoring. Mobile but straightforward cameras are usually used in vehicle-based systems. In intelligent vehicles, smart cameras already play a significant role. Smart cameras already play an important role in intelligent vehicles. Embedded vision components monitor the vehicle environment and the driver's condition and attention inside the vehicle. Adaptive cruise control for intelligent vehicles based on smart camera technology or forward collision warning systems is two implementation examples.

- ❏ *Medicine*: Many medical applications will benefit from smart cameras. The RVT system from Leeser [232] is an FPGA-based smart camera that allows surgeons to see real-time retinal images with blood vessels highlighted during surgery. Clever Systems' PhenoScan system, for example, analyses mouse activity for drug tests and can be used to automate experiments. Smart cameras can be used to keep an eye on patients and medical staff. When dealing with narcotics in the United States, for example, more than one individual must be present, a rule that can be checked using computer vision.

[4] Metal-Oxide-Semiconductor
[5] Charge-Coupled Device

- ❏ *Entertainment and Smart Environments*: Robotics, medical imaging, and entertainment are only a few of the uses for smart cameras. In multimodal user interfaces, gesture recognition will be critical. Princeton [243] has created a gesture recognition system based on distributed smart cameras. Aghajan et al.[463] have used distributed cameras to build a gesture recognition device. The authors concentrate on combining gestural elements from various cameras to distinguish human gestures accurately. Smart cameras are used as reliable, intelligent sensors in robotics in general.
- ❏ *Machine Vision*: Machine vision, which applies computer vision methods to manufacturing, inspection, and robotics, has made extensive use of smart cameras. These applications also necessitate high processing speeds and visual techniques that are thus incorporated into the overall application.

2.4.3 Distributed system in video surveillance

Recent advances in computing, communication, and sensor technology are pushing the development of many applications [70]. Amount this application, intelligent video surveillance systems. These systems generally come with a relatively large amount of computational and storage resources, but these are spread, both spatially and topologically. Consequently, random access to a distant resource is costly in required network bandwidth, especially in wireless multi-hop networks. While this issue can be trivially addressed by copying all data and letting each node processing it separately, this does not solve the polynomially-increasing communication burden [274].

As an alternative solution, processing in a large-scale camera network can be organized in a truly distributed manner using smart cameras, thus providing enough processing power and storage capacity even for complex tasks. In a network of embedded smart cameras, while, from the system viewpoint, nodes are spread across the network, each sensor has local memory and storage, and it can perform processing operations onboard [274]. According to Valera and Velastin [426], a distributed video surveillance system with distributed storage and network communication could enable real-time intelligent monitoring, improve the reliability and stability of video surveillance systems, and meet a variety of centralized system needs. For this reason, a distributed intelligent multi-camera video surveillance systems are becoming more common in today's surveillance systems.

In a distributed framework, each of the cameras in the network acts as an autonomous entity or agent that can record and analyze the data locally. However, data access costs depend on the physical and topological distances between network nodes, which is an inherent property of distributed systems in general. So, when performing processing operations in a network of embedded smart cameras, we can move the computational steps to the edge of the network and leave each node to perform the required actions. Then each sensor exchanges only the proper information to its neighbours to finally reach a shared and global objective in a fully distributed fashion [274].

This distribution aims to distribute intelligence and, in particular, to move part of the intelligence closer to the edge network (i.e., to a camera). This rapprochement is explained by the following requirements [349]:

- ✎ *Speed*: Solving lower-level peripheral problems not only makes the processing chain parallel, but also allows central processing nodes to focus on more sophisticated analysis, like threat assessment.

- ✎ *Bandwidth*: Edge-level processing eliminates the need to transmit high-bandwidth and highly redundant data streams to central processing nodes.
- ✎ *Redundancy*: Increased reliability since the distributed computing platform can be reconfigured to handle component failures.
- ✎ *Autonomy*: The crucial feature of a truly distributed system. A large surveillance system is usually composed of various cameras widely dispersed over a large geographical area. To process video data streams asynchronously, different semantic levels of knowledge must be generated by different types of users.

2.4.4 Multi-camera coordination system

Multi-camera surveillance systems have received significant attention in both research and industry. This type of system is defined by Prabhu Natarajan et al. in [298] as a mechanism allowing several heterogeneous cameras to capture videos, analyse, share knowledge, perform calculations, and control actions collaboratively to submit a surveillance service.

"We define Multi-Camera Coordination and Control (MC3) as a mechanism by which multiple heterogeneous cameras: (i) capture and analyse their videos, (ii) collect and fuse the shared knowledge about the surveillance environment from their neighbouring camera nodes through communication, (iii) compute and execute optimal control actions in order to perform a desired surveillance task in a collaborative manner." (Natarajan et al., [298])

There are complex situations in many camera systems. For example, when one or more cameras detect an intruder is suspected, in this case, the cameras have to follow the intruder continuously in the observed area. This means the cameras must be controlled to track the intruder at high resolution to get more information. Also, cameras that focus on tracking the intruder must transmit their data to their neighbouring cameras to be ready to track a potential intruder if he enters their field of view. All these operations can only be carried out if there are proper coordination and control among the camera nodes. Analogous to a human security team, where each security personnel communicates with other security personnel during any policing operation, surveillance cameras are also required to communicate and coordinate with each other [298]. Figure 2.11 shows an illustrative example of the complex coordination between several cameras in a multi-camera system.

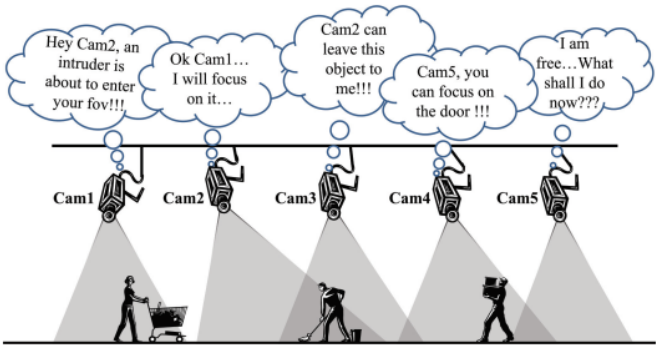
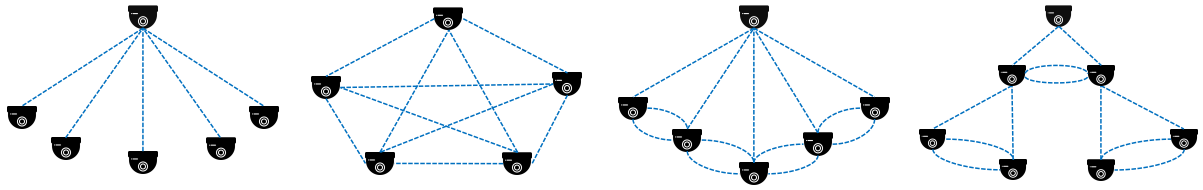


FIGURE 2.11: An example scenario for multi-camera coordination

2.4.5 Multi-camera coordination architectures

In this part, we present an overview of the fundamental multi-camera coordination and control architectures that have been introduced in the literature. According to [298] and [313], there are four basic architecture models, which are: centralized, distributed, hybrid and hierarchical (see Figure 2.12).



(A) Centralized (B) Distributed (C) Hybrid (D) Hierarchical

FIGURE 2.12: Multi-camera system architecture

2.4.5.1 Centralized architecture

In a centralized architecture, multi-camera data aggregation and analysis are performed at the BS. In this type of architecture, no autonomous decisions or processing will be performed by any distributed cameras. The BS is usually responsible for *(i)* fusion of data from multiple cameras on the network, *(ii)* information assimilation and, *(iii)* making decisions and control the cameras [298]. Several approaches have used this architecture, such as Lu and Payande [258], Zhang et al. [507], He et al. [181], and Lin et al. [246]. All the information that is exchanged among the distributed cameras goes through the BS. Figure 2.12a shows the connection setup of centralized architecture.

2.4.5.2 Distributed architecture

In contrast to centralized architectures, a distributed architecture consists of independent camera nodes that operate autonomously by exchanging information with their neighbouring camera nodes to perform the desired surveillance task (see Figure 2.12b). In this architecture, smart cameras capable of sensing, processing, and communication are usually employed in a distributed architecture.

The functional architecture of the centralized and distributed architecture components for a typical surveillance application is represented in Figures 2.13a and 2.13b, respectively.

In a centralized architecture, as shown in Figure 2.13a, a camera performs object detection, tracking, and classification and sends the results to their BS. The BS then fuses the results from different camera nodes and decides which targets will be tracked by which camera nodes. In a distributed architecture, each camera node performs both application-specific and coordination tasks. As shown in Figure 2.13b, each camera performs object detection, tracking, classification as specific tasks, data fusion, information assimilation, and decision-making as coordination tasks. In the latter, all camera nodes share their own (local) knowledge about the surveillance environment with their neighbouring nodes and develop the global or near-global knowledge of the surveillance environment. Thanks to this sharing, each camera tries to make local decisions that improve the overall surveillance objective.

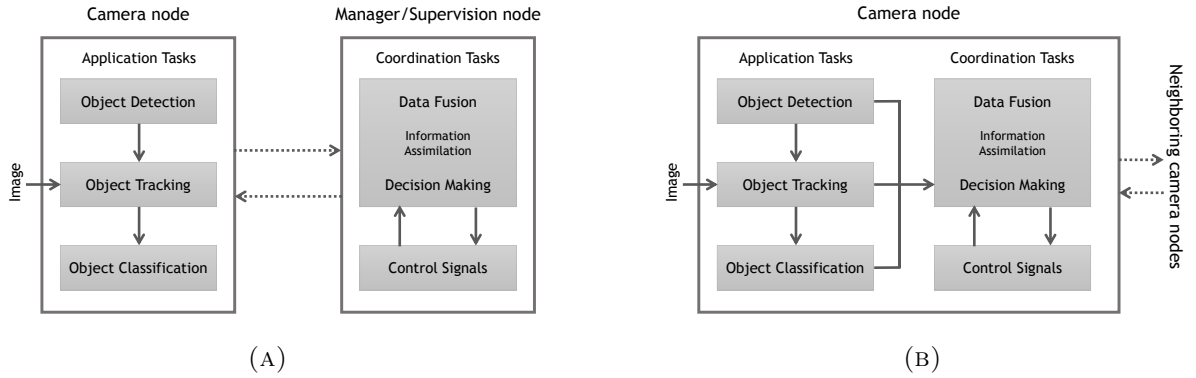


FIGURE 2.13: The functional architecture of components in (a) centralized and (b) distributed architectures

2.4.5.3 Hybrid architecture

The combination of centralized and distribution architecture functionality forms the hybrid architecture (see Figure 2.12c). In this architecture, cameras perform low-level functions such as object detection, tracking, and classification and submit the results to the central processing unit. The cameras make certain decisions at their level based on their own knowledge, while the higher-level decision remains up to the BS. The BS collects data from their cameras and fuses them to obtain useful information or make high-level decisions. This architecture combines the advantages of centralized and distributed architecture, which motivated each of Prati et al. in [334] and Diratie et al. [125] to use it as an architectural alternative to conventional architectures.

2.4.5.4 Hierarchical architecture

Hierarchical architecture can also be referred to as a multi-level architecture where the level of decision-making depends on the camera hierarchy level, i.e., the cameras in the upper layers control the cameras in the lower layers in a hierarchical way (see Figure 2.12d). Among the approaches using this architecture, we cite: Matsuyama and Ukita [275, 425], Bamberger et al. [71] and Kulkarni et al. [227].

In the end, we summarize the advantages and disadvantages of these architectures in Table 2.2.

2.4.6 Application of distributed multi-camera cooperative in IVSS

2.4.6.1 Multi-camera-based IVSS

The first research on the multi-camera system was carried out with fixed cameras. The cameras' resolution was of the low type due to their wider angle of view to cover large areas. The discovery of smart cameras made a significant contribution in object tracking, on-board processing, and obtaining detailed information about the image captured from the scene. In distributed smart cameras, information is shared between individual cameras with distributed detection and processing capability in a smart camera network. Ubiquitous smart cameras create autonomous and adaptive smart camera operations, facilitating use and operation in different application

TABLE 2.2: Advantages and disadvantages of camera architectures (from [313])

Architecture	***Advantage***	***Disadvantages***
Centralized	• Easy to control • Less collaboration among the cameras • Simple hardware arrangement	• Bottleneck communication path. • Higher computational capability • No redundant path
Distributed	• No bottleneck communication path • Many redundant paths. • Less computational time.	• There is a collaboration among cameras
Hybrid	• Combined the benefits of both centralized and distributed architecture. • Faster processing and communication	• Ineffective in large-scale cameras
Hierarchical	• Scalable network • Redundancy communication path • Better monitoring and control	• Calibration of cameras is required • Consumes a lot of time

areas. In the literature, several studies have focused on the multi-camera network, especially in surveillance systems. For the rest, we summarize in Table 2.3 a brief overview of the literature on surveillance systems based on a multi-camera network.

TABLE 2.3: Summary of recent studies on the application of multi-camera in surveillance systems (extended from [313])

Refs	***Years***	***Description***
[7]	2009	This work examined the variable technologies under sensor and surveillance systems for a vast area network. It described the modalities required for selecting, planning, and fusion data obtained from a sensor in a multiple sensor environment.
[377]	2009	Sheikh et al., in their work, describe the features of tracking objects in the multi-camera system based on changes in shape, movement, and size. The object's shape, motion, and size are not constantly fixed, and object tracking has been described as a region-tracking problem. The region is seen as a 2D image plane display of an object.
[12]	2009	Aghajan book describes the basic multi-camera configurations of image development and camera model, the geometry of the stereo vision and camera matrix, the projective transformation form, and the construction of n-cameras. The other features discussed include detection and matching and basic evaluation algorithms in multi-camera systems.
[391]	2009	The authors create a framework for heterogeneous cameras with overlapping FoV of the same spatial location in this work. The authors make each camera homogenised output to avoid the need to supervise the camera or create a constraint for the camera parameters.
[216]	2010	This work focused on the study and elucidation of object responses, detection, and tracking to perceive the scene's look. Also, this paper examines the large-area control of cameras used to track a moving object in a multi-camera system.
[370]	2010	Seema and Reisslein carried out a review of hardware and software on wireless video sensor platforms for surveillance. Based on their investigation, they produced a new wireless video sensor network with a low-cost frame design.

Refs	***Years***	***Description***
[84]	2011	It described a real-time multi-camera-based vehicle-tracking system for tunnel surveillance. An algorithm (i.e., inter-camera matching) was used to calculate the similarity of projections between vehicle images.
[409]	2012	This paper discusses the different visual sensor networks available and image compression methods, video coding, and computer vision applicable to those platforms.
[48]	2012	The authors present a real-time intelligent surveillance system that detects and tracks moving objects and automatically warns of suspicious activities such as monitoring unidentified objects and restricted areas. The proposed approach offers different views of multi-cameras with significant overlap between their FoVs, as an association between them to resolve the occlusion in a new and straightforward way.
[441]	2013	D'Orazio and Guaragnella further studied the importance and presented a survey of the automatic event detection functionalities developed for multi-camera systems, focusing on open problems restrict computer vision methodologies to commercial applications.
[132]	2015	This paper reviews the critical computer vision and pattern recognition technologies, including multi-camera calibration, computation of the topology of camera views, multi-camera tracking, object re-identification, and multi-camera activity analysis with a detailed description of technical challenges and comparison of various solutions.
[313]	2020	The authors of this survey analyse multi-camera systems in terms of physical arrangements, calibrations, algorithms, and their advantages and disadvantages. This system is examined in four application areas of multi-camera systems (surveillance, sports, education, and cell phones). The authors also discuss the aspect of calibration and architecture in multi-camera formation.
[256]	2020	The first use of multi-camera collaboration for image validation as precision compensation of lightweight models in video compression is from Y. Liu et al. They present a video pre-processing strategy for wireless surveillance systems using lightweight AI and multi-camera collaboration.

2.4.6.2 Co-operative camera systems

Surveillance of large areas requires using a large number of cameras to cover as much area as possible and achieve good performance in the automatic surveillance operation. Consequently, the need to coordinate information between cameras becomes an important issue. With the development of smart cameras, researchers' attention has shifted to self-control and collaboration between distributed smart cameras that execute the vision algorithms we have previously discussed and information from neighbouring cameras. In the rest, we highlight the main works in this area.

The co-operative camera network (CNN) [323], is an indoor application surveillance system consisting of a network of nodes. Each node consists of a camera connected to a PC and a central console to be used by the human operator. The system signals the presence of a visually tagged individual inside the building, assuming that human traffic is sparse (an assumption that becomes less valid as crowd levels increase). The objective of the system is to monitor potential shoplifters in department stores.

A surveillance system for a parking application is described in [284]. The system architecture consists of one or more static cameras and one or more PTZ cameras. At first, static cameras detect and track the target. A PTZ camera is activated to capture the target's high-resolution video if the target has been selected. In this system, the Mahalanobis distance and Kalman filters are used to fusion data for multi-tracking and tracking purposes, respectively.

Nguyen et al. [304] present an interesting example of a multi-tracking camera surveillance system for indoor environments. The system is a network of camera processing modules, each consisting of a computer-connected camera and a control module. Each camera-processing module realises the tracking process using Kalman filters. The authors develop an algorithm, which divides the tracking task between the cameras by assigning the tracking to the camera, which has better visibility of the object, taking into account the occlusions. The idea is interesting because it shows a technique that exploits distributed processing to improve detection performance.

Yeh et al. [484] proposed a cooperative dual-camera system for frontal surveillance. In this system, the authors used two cameras to track the object correctly. The two cooperative cameras accurately analysed the shape of the human body while the other camera predicted the movement, position, and height of the human body. Experimental results show that the proposed algorithm performed well when considering the clarity of close-up views for both multiple and single objects.

To improve real-time tracking, Shuai Zhao et al. [510] propose a dynamic camera collaboration framework based on the community concept to detect overlapping areas. This mechanism allows through the Clique Percolation Method (CPM) to optimise the number of active groups.

Niccolò et al. [61] propose a new decentralised approach to reconfiguration the heterogeneous camera network, where each camera dynamically adapts its parameters and position to optimise the coverage of the scene. In the proposed decentralised reconfiguration policies, the cameras can locally control the state of their neighbourhood and dynamically adjust their position and parameters based on a cooperative approach.

To achieve online real-time monitoring of moving targets in smart manufacturing scenes, Anhu et al. [236] propose a cooperative camera surveillance method based on the coarse-fine coupling boresight adjustment principle. The master-slave camera model is established by combining a high-resolution camera with a wide field camera. The master camera has a wide field of view to assist in target guidance and a slave camera integrated with a Risley prism pair to achieve efficient and high precision line-of-sight pointing and high-resolution imaging for return on investment.

As illustrated, in a distributed multi-camera surveillance system, it is important to know the topology of the links between the cameras that make up the system to recognise, understand, and follow an event that may be captured on one camera to follow it in other cameras. Most of the multi-camera systems that have been discussed in this review use a calibration method to compute the network camera topology. Additionally, most of these systems attempt to either combine tracks from the same target simultaneously in different camera views or distribute the system load among all adjacent cameras to achieve a real-time system.

2.4.7 Requirements & challenges

As mentioned above, many works in the literature have identified different surveillance problems using multi-camera systems, while some propose solutions to others' issues. Given recent

advances in distributed multi-camera coordination systems in research, we list several trends and challenges identified from our review.

At present, where there are multiple targets to be tracked, surveillance challenges using multi-camera systems require multi-camera coordination. The number of targets observed by a specific camera without losing the quality of the image captured is not estimated. Mainly when there is a lower camera/target ratio. If this is the case, i.e. if the number of targets increases with the number of cameras, it can exponentially increase computing time, which degrades the entire surveillance system's performance. This problem is particularly aggravated if the system itself is poorly designed.

A computational and processing problem, in particular for real-time applications, is another problem in multi-camera systems. Surveillance is a real-time activity that requires a lot of processing power from the cameras on a surveillance network. Currently, the processing capabilities of smart cameras are still insufficient when it comes to many cameras. Even if smart cameras can process faster, the communication medium still limits the data transfer rate due to the traffic load from one camera to another.

A combination of different disciplines should achieve a large-scale distributed intelligent surveillance system: computer vision, telecommunications, and systems engineering are needed. Therefore, it is essential to create a framework or methodology to design distributed extended surveillance systems, from requirements generation to creating design models by defining functional and intercommunication models as is done to create real-time distributed systems.

The main prospective challenge is to develop a large-scale distributed multi-sensor monitoring system with robust, real-time computing algorithms capable of operating with minimal manual reconfiguration on variable applications. These systems must be adaptable enough to automatically adjust to and cope with changes in the environment, such as lighting, scene geometry, or scene activity. The system must be sufficiently scalable, based on standard hardware, and exploit plug-and-play technology.

There are obstacles such as pillars, walls, and barriers in many real-world surveillance environments that mask the FOV of some or maybe even all of the cameras. This may also be attributed to privacy concerns when tracking those environmental regions. It is, therefore, highly inefficient for cameras in such environments to persistently track the targets observed.

In cooperative multi-camera systems, the cameras must form a group to solve a surveillance system task. Forming the right group at the right time is another critical issue. Groups must include additional sensors specific to a task. For example, IR cameras must create a group and collaborate when there is less or no lighting or motion detectors to activate cameras only in case of movement. The problem of forming groups between several cameras is complex and deserves to be studied in the future.

Distributed multi-camera coordination is complex and critical in power-consuming camera networks. Therefore, system algorithms must be optimised to consume less energy. This problem needs more attention from system researchers.

Distributed multi-camera coordination systems can be seen as some form of sensor network with different characteristics: i) fewer nodes; ii) higher computing, communication, and power resources per node; and iii) in most cases stationary nodes with established networking [350]. The unique characteristics of distributed smart cameras as sensor networks leads us to some important problems. The higher computing and communication rate leads us to consider more

sophisticated distributed computing services, such as task migration, load balancing and automated camera scheduling. These are also important issues for these systems.

2.5 What next?

Over the last two decades, the IoT's development has witnessed its vast potential in that physical environments can be equipped with intelligent things integrated with new information and communication technologies for seamless cyber-physical interaction. However, nowadays, objects are no longer limited to personal objects such as smartphones, tablets, and smartwatches. Today's objects, on the contrary, include large-scale intelligent things or sensors embedded in our environment (such as houses, vehicles, buildings, or infrastructure) connected by a gateway device that transmits data to the processing centre. Among these things, we find visual captures or cameras. These captures are widely deployed, and their generated data represents more than 75% of IoT traffic. Thanks to their volumetric and content generation of visual data, video surveillance systems require significant storage resources, transmission bandwidth, and energy consumption.

To increase the flexibility and reduce the cost of deploying video surveillance systems, smart cameras together with a fusion of cloud, fog, mist, and edge computing paradigm and all Internet of Things (IoT) they formed the next future generation of video surveillance systems which called it "Internet of Video Thing or IoVT". IoVT is a part of the IoT capable of efficiently processing large volumes of data, such as images and videos. Compared to conventional systems, VSS in an IoVT framework provides multiple layers based on new computing paradigms such as edge, mist, fog, and cloud computing as an infrastructure of communication and decision-making by capturing and analysing rich contextual and behavioural information. IoVT aims to develop a more efficient, flexible, and cost-effective video surveillance system adapted to the new requirements of smart cities in terms of the safety and security of citizens in public and private places.

From the discussion above, we can define the IoVT as:

"A network of visual capture or smart cameras, uniquely identified without ambiguity, operates in an intelligent environment. These smart cameras can interact and communicate with each other and/or with other IoT objects and humans using Information Communication Technologies (ICTs) for distributed processing, data exchange/sharing, and increase system autonomy".

IoVT objects are different from IoT objects. They require more storage space because of the massive amount of data they generate, more computing power to process their complex data and more energy with higher bandwidth for supporting data traffic. Table 2.4 summarises the main characteristics of IoT and IoVT. Real-time deployment scenarios include industrial IoT, smart cities, smart hospitals, smart agriculture, and smart homes.

Two concepts have appeared in recent years with "IoVT": Internet of Multimedia Things (IoMT) [24, 345, 517] and Internet of Surveillance Things (IoST) [151].

Concerning the "IoMT", Sheeraz et al. in [24] define it as :

"The global network of interconnected multimedia things which are uniquely identifiable and addressable to acquire sensed multimedia data or trigger actions as well as possessing capability to interact and communicate with other multimedia and not multimedia devices and services, with or without direct human intervention" (Sheeraz et al., [24])

TABLE 2.4: Comparison of IoT and IoVT

Parameters	**IoT**	**IoVT**
Quantity of data generated	*Linear*	*Ultra-big*
Processing	*Low*	*Excessive*
Storage	*Low*	*Massive*
Bandwidth	*Low*	*High*
Delay	*Tolerant*	*Sensitive*
Energy	*Low*	*High*
Interoperability	*High*	*High*
Scalability	*High*	*High*

While "IoST" appears in a single article [151], but without giving a definition to any information on this concept.

From the above definitions, we can say that the IoMT covers the problems of all the multimedia sensors and the massive data generated by them, and IoVT focuses only on visual sensors that represent both cameras and their types and video or image data. In contrast, IoST focuses on the application of video surveillance in the IoT. Finally, we can propose the hierarchy of these concepts as represented in the following diagrams (Figure 2.14):

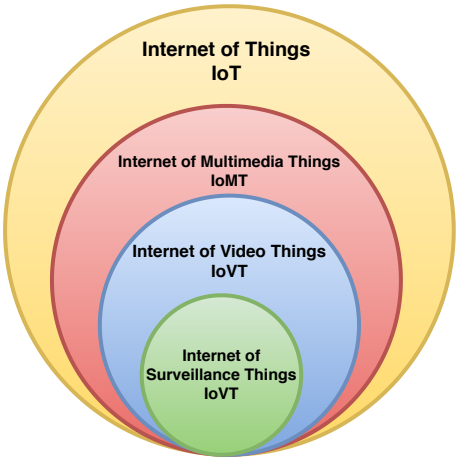


FIGURE 2.14: The proposed hierarchy of IoT, IoMT, IoVT, and IoST

2.6 Conclusion

In this chapter, we have presented a review of the development status of intelligent distributed surveillance systems, including a brief historical summary of surveillance systems (section 2.2) and a review of current image processing techniques used in different modules that are part of surveillance systems (section 2.3). In section 2.4, we have defined the reason for using distributed intelligence and a cooperative distributed system in CCTV systems. Before concluding this chapter, in section 2.5, we have presented a vision of the future of video surveillance.

CHAPTER 3

INDEXING DATA TECHNIQUES

Chapter contents

3.1 Introduction

The emergence of the IoT allows many objects to interact seamlessly and with other resources on the network. However, the massive amount of data generated by these objects and their distinctive characteristics make the use of traditional database systems inadequate as a solution. For this reason, IoT requires new solutions for large-scale applications to enable fast and efficient search and discovery of their data and services. Therefore, IoT requires scalable and efficient distributed indexing solutions for large-scale distributed IoT networks.

In the IoT, the objects to be indexed are often more complex than simple vectors. For this reason, data indexing has acquired more requirements. Indexes in multidimensional space are more rigid because of their strong dependency by type or, more specifically, by their geometric properties. This dependency makes the implementation of a flexible structure for a reliable system very difficult. Consequently, the focus of indexing has partly moved from multidimensional spaces to metric spaces.

The metric space has been proposed as a universal abstraction for large data volumes, or Big data [270]. Among the main features of this space, it does not require the intrinsic structure of the data, but only a distance function, with the properties of non-negativity, symmetry and triangular inequality of pairs of data points [353]. The most important advantage of indexing in metric space is that more data types can be indexed because the approach is based only on calculating distances between objects and not on their content [498]. In other words, the metric space is more flexible and simpler than the multidimensional space, allowing us to create an index structure that can handle any data.

In the previous chapter, we have discussed our application area. This chapter will focus on data generated by the IoT, including video surveillance systems as an important data source. In section 3.2, we present the general cycle of the data management process. Then, we focus on the management and indexing of large data in the field of IoT. Section 3.3 presents the necessary elements to understand data indexing and searching in metric spaces as a data abstraction space. Furthermore, in section 3.4, we present an in-depth analytical study on existing indexing techniques in the literature.

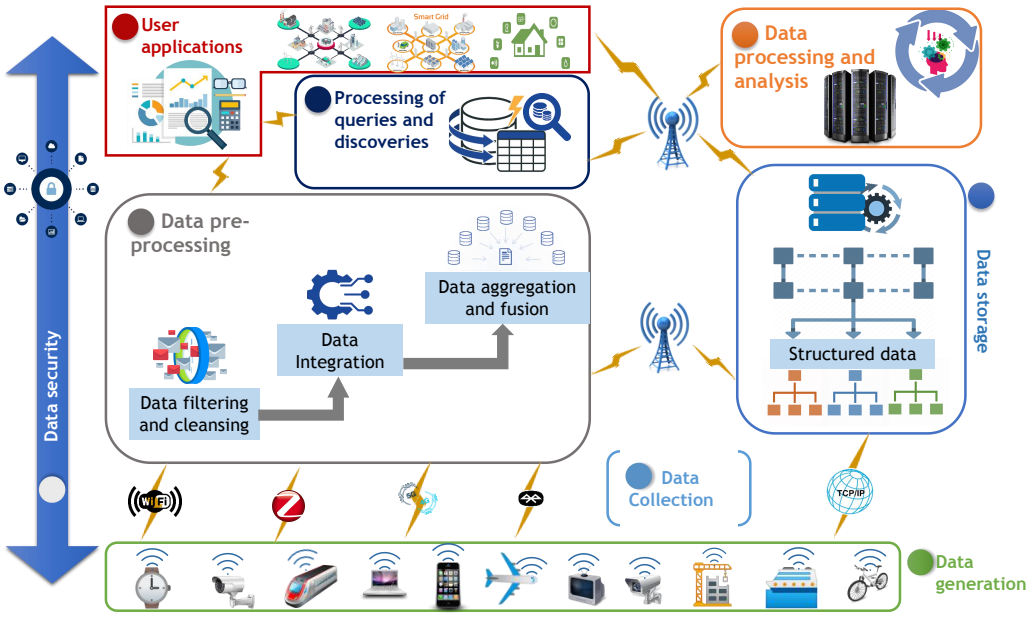


FIGURE 3.1: Life-cycle of data management

3.2 Data management

The basic logical and physical structure of the data management can be identified from different angles. Figure 3.1 presents the main components of management, which can be summarized in six main modules: *data collection*, *data pre-processing*,*storage/update – data archiving*, *data processing and analysis*, *query and discovery processing*, and *data security and confidentiality*.

- ***Data collection***: The first step in the data management process is the collection of data by the different technologies deployed in the network (such as sensor networks (SNs), unmanned aerial vehicles (UAVs), vehicle ad hoc networks (VANETs), etc. [154] with different strategies based on specific criteria such as the availability of financial resources, the amount of data, the collection period (temporal or modal), etc.

  Temporal data collection involves collecting data from connected objects at specified intervals, while modal data collection involves collecting data about specific elements [8].

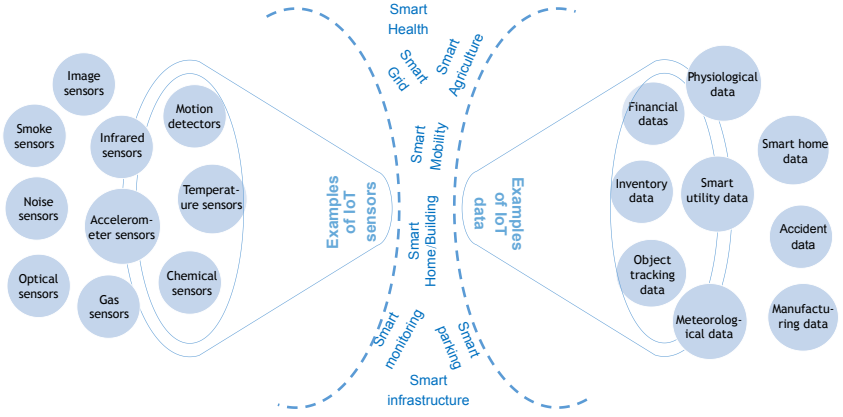


FIGURE 3.2: Big IoT data sources

- ***Data pre-processing***: Data will come from different sources with varying formats and structures, especially in the IoT age (see Figure 3.2). To make high-quality decisions, high-quality data is necessary. Therefore, it is necessary to check and control the data quality before analysis [432]. For this reason, data pre-processing provides a mechanism to transform the raw data collected by the previous module, which may be incomplete or inconsistent and contain many errors or unusable parts, into a format that is understandable and ready for the analysis process [385]. The pre-processing data phase prepares the raw data for further processing [282] and transforms it into a format that makes processing more efficient [490]. According to studies conducted in the literature, there are three main processes for data pre-processing:

  - *Data filtering and cleaning* : The process of data filtering and cleansing is a crucial step to achieve good results. This process is used to identify and remove inaccurate, incomplete, redundant and inconsistent data [229]. In this way, the storage cost is reduced, and data is transmitted faster [364].
  - *Data Integration* : Data integration is defined as a set of techniques used to provide a single or unified view of data distributed over different sources [14]. Integrated data is used in the aggregation phase for a more detailed summary.
  - *Data aggregation and fusion* : As the size of IoT data is important, aggregation techniques play a key role in improving the global efficiency of IoT networks [472]. Data aggregation's fundamental objective is to efficiently aggregate and collect massive data packets before being transmitted to the base station for storage. Data aggregation aims to reduce power consumption, improve network lifetime, and increase data accuracy [342]. Figure 3.3 shows a classification of data aggregation techniques according to [331].

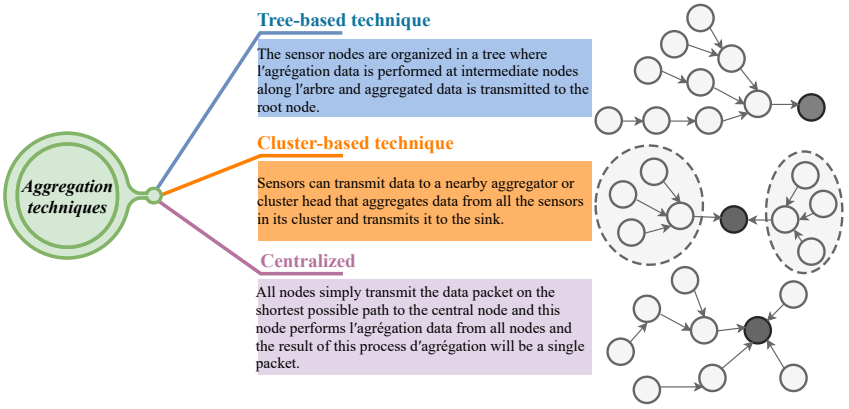


FIGURE 3.3: Classification of data aggregation techniques (from [331])

- ***Storage/Update – Data archiving***: In the management of IoT data, it is difficult to obtain optimised and efficient query execution results, especially when the data in question is massive and large-scale. In the data management life-cycle, the storage phase ensures the efficient storage and organisation of massive data and the continuous updating of data with new information. It becomes available for further processing and analysis and more in-depth queries. Archiving refers to the long-term offline storage of data that is not immediately necessary for system operation. The heart of centralised storage is the deployment of storage structures that adapt to the different data types and the frequency of data entry [8].

Indexing techniques are gaining popularity as solutions for organising data to complete the storage phase, as they improve performance and facilitate the efficient retrieval of large data sets.

- ❖ *Indexing* : Indexing of heterogeneous and dynamic big IoT data requires distributed, efficient and scalable mechanisms that can provide fast data access and retrieval to meet user demands [141]. Building and storing indexes is prohibitively expensive, both in terms of storage and processing. However, distributed dynamic indexing has been recognised as an essential step in the storage phase.

- ❑ ***Data processing and analysis***: Data processing is a fundamental phase in the lifecycle of data management. This step is defined as a conversion method/process used to extract and convert stored data into valid and relevant information [214]. The data processing task is recognised as a complex, resource, and energy-intensive procedure compared to other data management lifecycle processes. Several key factors, namely control this process :
  - ✎ Sensor type
  - ✎ The state of the environment
  - ✎ Measurement and data quality control
  - ✎ The efficiency of data storage and organisation
  - ✎ Data processing techniques
- ❑ ***Query and discovery processing***: End users (machines, services, or human users) in IoT applications can discover and query a particular data/object with specific characteristics [141]. If the characteristics are known in advance, the user can search for exact queries or search for objects similar to a corresponding object if the user does not know in advance the characteristics of the data/object in question (similarity or inexactness queries). In this framework, query processing efficiency is highly dependent on the efficiency of the organisation or indexing structure of the data in the database.
- ❑ ***Data security***: The security (integrity, confidentiality, availability, and accountability) of data is a process that touches every step of the data management life-cycle. Data security and privacy are essential for IoT networks because some objects that generate data may be the property of particular entities, data intended for military or government applications, or private individuals, such as data generated by medical devices [8].

In our context of work, we are interested in "*Storage/Update – Data archiving*" and "*Query and discovery processing*" of the data management life-cycle and, in particular, in the massive indexing of data in the context of the IoT.

3.3 Foundation of metric indexing: Theoretical background

Images in our context go through the feature extraction process, so every image is represented as a K-dimensional feature vector. The basic assumption is that the two images are similar according to a distance metric if the corresponding feature vectors are similar. The metric space approach is very important in building effective indexes for similarity searching. The most important advantage is that many data types can be indexed because the metric space does not make any requirements on their content or the intrinsic structure of the data, but only a function of distance [271, 498].

3.3.1 Metric space

In the context of IoT data, "variety" is relatively less studied. To conquer "variety", one can first find a universal abstraction covering different data types and then build a data management and analysis system based on universal abstraction characteristics. According to J.-L. Verley [431], the notion of metric space, was introduced by Fréchet and later developed by Hausdorff. Metric space has been proposed as a universal abstraction for data [270]. Among the main features of this space, it does not require the intrinsic structure of the data, but only a distance function, with the properties of non-negativity, symmetry and triangular inequality of pairs of data points [353].

Definition 3.1 (Metric Space)**.** A metric space $\mathcal{M}$ is a two-tuple $(\mathcal{X}, d)$, where $\mathcal{X}$ is a set of valid objects and $d(\cdot,\cdot)$ is a distance function to measure the "similarity" between two objects or two elements of $\mathcal{X}$. In particular, the distance function present in (3.1) has four properties:

$$d : \mathcal{X} \times \mathcal{X} \Longrightarrow \mathcal{R}^{+}$$

1. Non-negativit :
$$\forall x, y \in \mathcal{R},\ x \neq y \Longrightarrow d(x, y) \geq 0 \tag{1}$$
2. Symmetry :
$$\forall x, y \in \mathcal{R},\ d(x, y) = d(y, x) \tag{2}$$
3. Identity :
$$\forall x, y \in \mathcal{R},\ d(x, y) = 0 \Longrightarrow x = y \tag{3}$$
4. Triangle inequality :
$$\forall x, y, z \in \mathcal{R},\ d(x, y) + d(y, z) \geq d(x, z) \tag{4}$$

3.3.2 Distance measurements of metric space

This section provides some kinds of distance functions used in practice on different types of data. Distance functions can be adapted to a specific application or field of application. An expert then specifies them. However, the definition of a given distance function is usually not limited to a single type of query.

According to [219], distance functions can also be classified according to their computational cost. The simplest ones are linear in the size of the object, which is optimal. Others are already quadratic, such as the distances between sets or the editing distance described above. But others, maybe even more complex. In this thesis, we are not directly interested in the distances used, but we take this constraint into account and minimise distance calculations.

Minkowski distances

Minkowski distances represent a complete family of metric functions, referred to as L_p, because the individual instances depend on the parameter p. These functions are defined on n-dimensional vectors of real numbers, as indicated in definition 3.2:

Definition 3.2 (Minkowski Distances)**.** The Minkowski distance of order p (where p is an integer) between two points $X = (x_1, \cdots, x_n) \in \mathcal{R}^n$ and $Y = (y_1, \cdots, y_n) \in \mathcal{R}^n$ is defined as:

$$L_p(X, Y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p} \tag{5}$$

Minkowski distance is typically used with p being 1 or 2, which correspond to the *Manhattan* distance and the *Euclidean* distance, respectively. In the case of p reaching infinity ($p \to \infty$), we obtain the *Chebyshev* distance (is also called the *maximum* distance or the $infinite$ distance).

- *Manhattan* distance (L_1) is described simply as follows:

$$L_1(X, Y) = \sum_{i=1}^{n} |x_i - y_i| \tag{6}$$

- *Euclidean* distance (L_1):

$$L_2(X, Y) = \sqrt[2]{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{7}$$

- *Chebyshev* distance (L_∞)

$$L_\infty(X, Y) = \max_{i=1}^{n} |x_i - y_i| \tag{8}$$

3.3.3 Similarity queries in metric space

In the current IoT era, exact match recovery, typical of traditional databases, is neither feasible nor meaningful for large-scale data and emerging data types in the Big IoT era. The reason is that modern digital collections' ever-expanding data suffer from a lack of structure and accuracy. For this reason, what constitutes a match with demand is often different from what is involved in more traditional and well-established fields.

As an alternative solution, a new search paradigm is introduced, the latter being known by Similarity Queries. This search paradigm measures a query object's proximity, similarity, or dissimilarity to objects stored in a database to be searched. Roughly speaking, the objects close to a given query object form the set of answers to the query. The mathematical concept of metric space [212] provides a useful abstraction for proximity [498].

The problem of indexing metric spaces can be defined according to [126] as follows:

Problem 3.3.1. *Let $\mathcal{O}$ be a domain, $d(\cdot, \cdot)$ a distance function defined on $\mathcal{O}$, and $\mathcal{M}(\mathcal{O}; d)$ a metric space. Given a set $\mathcal{X} \subseteq \mathcal{O}$ of n elements, preprocess or structure the data so that similarity queries can be answered efficiently.*

A similarity query is defined explicitly or implicitly by a query object q and a constraint on the form and extent of proximity required, typically expressed as a distance. The response to

a query returns all objects that satisfy the selection conditions, presumed to be those close to the given query object. In the following, we define the two best known and most used types of similarity queries.

❑ **Range Query**

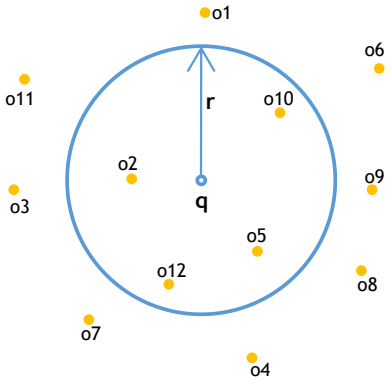
The query $\mathcal{Q}(q, r)$ is specified by the query object $q \in \mathcal{O}$, and the query radius r and retrieves the subset of all elements that are at a distance less than r from a given element q. The following definition can formally describe this:

Definition 3.3 (Range query)**.** Let $\mathcal{M}(\mathcal{O}; d)$ a metric space, $q \in \mathcal{O}$ a request point, and $r \in \mathcal{R}^+$ the query radius:

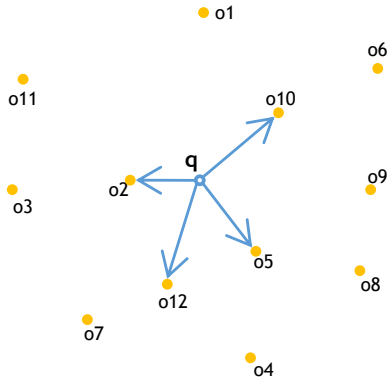
$$\mathcal{Q}(\mathcal{O}, d, q, r) = \{o \in \mathcal{X} : d(q, e) \leq r\} \tag{9}$$

A range query is simply determining the subset of elements in a metric space belonging to the extension of a ball. Figure 3.4a shows an example of the range query. Another example of a range query in a geographic application, carried out with a mapping system: *"Q: Give me all the museums located within a radius of 2 km from my hotel"*.

Remark 3.3.1. A query object q does not necessarily have to belong to the object dataset, i.e., $q \notin \mathcal{X}$. On the other hand, the query object must be included in the $\mathcal{O}$ metric space domain, i.e., $q \in \mathcal{O}$. This statement applies to all possible types of queries.



(A) Range query $\mathcal{Q}(\mathcal{O}, d, q, r)$



(B) kNN query $\mathcal{NN}(\mathcal{O}, d, q, k)$

Remark 3.3.2. When the search radius is equal to zero, the range query $\mathcal{Q}(\mathcal{O}, d, q, 0)$ is called a point query or exact match. In this case, we look for identical copies of the query object q. Usually, this type of query is used in deletion procedures when we want to locate objects to remove them.

❑ k **Nearest Neighbor query (**k**NN)** In the nearest neighbours query, we look for the k objects most similar to a given element q. The query is nothing more than a reference element q, and the number of elements searched k. The set of answers is never empty, except when the set is empty or trivial exceptional cases. In addition, the size is defined beforehand by the user. The elementary version of this query finds one nearest neighbour (1NN), the object closest to the given query object.

The following definition can formalise this type of query.

Definition 3.4 (k Nearest Neighbor Query)**.** Let $\mathcal{M}(\mathcal{O}; d)$ a metric space, $q \in \mathcal{O}$ a request point, and $k \in \mathcal{N}$ the required number of answers.

$$\mathcal{NN}(\mathcal{O}, d, q, k) = \{\mathcal{S} \in \mathcal{X}, |\mathcal{S}| = K \wedge \forall x \in \mathcal{S}, y \in \{\mathcal{X} - \mathcal{S}\} : d(q, x) \leq d(q, y)\} \quad (10)$$

Here again, it is a query of determining a subset in extension from a ball. The difference compared to a range query seen previously is that the radius r is initially unknown. Therefore, it will be determined afterwards as the distance to the k^th element of the answer. Figure 3.4b illustrates an example of the k query of the nearest neighbour.

Remark 3.3.3. In the case where several objects are at the same distance from the request, the choice is made randomly [126, 498].

Remark 3.3.4. If the collection to be searched consists of fewer than k objects ($|\mathcal{X}| < k$), the query returns the whole database ($|\mathcal{S}| = |\mathcal{X}|$) [126, 498].

3.3.4 Basic partitioning policies

Partitioning is one of the essential concepts of any storage system. It aims to partition the search space into subgroups, such that only some of these groups are searched once a query is given. Uhlmann [424] identified two basic partitioning schemes, *ball partitioning* and *generalized hyperplane partitioning*. In the following, we briefly define these techniques.

❑ **Ball partitioning**

A ball is a topological notion that generalises a disc in the Euclidean plane and a sphere in space. Ball partitioning breaks the set $\mathcal{O}$ into subsets $\mathcal{X}_1$ and $\mathcal{X}_2$ using a spherical cut with respect to $p \in \mathcal{O}$, where p is the pivot, chosen arbitrarily. Let d_m be the median of $\{d(o_i, p), \forall o_i \in \mathcal{X}\}$. Then all $o_j \in \mathcal{X}$ are distributed to $\mathcal{X}_1$ or $\mathcal{X}_2$ according to the definition 3.5:

Definition 3.5 (Closed Ball)**.** Let $\mathcal{M}(\mathcal{O}; d)$ a metric space, a set $\mathcal{X} \subseteq \mathcal{O}$ a pre-treated or structured data, $p \in \mathcal{X}$ a pivot object, and $d_m \in \mathcal{R}^+$ the radius of coverage. $\mathcal{B}(\mathcal{O}, d, p, d_m)$ defines a ball that partitions the space into two parts:

$$\begin{aligned} \mathcal{X}_1 &= \{o \in \mathcal{X} : d(o, p) \leq d_m\} \\ \mathcal{X}_2 &= \{o \in \mathcal{X} : d(o, p) > d_m\} \end{aligned} \quad (11)$$

In this definition, $\mathcal{X}_1$ represents the interior of the Ball and $\mathcal{X}_2$ represents the exterior of the Ball.

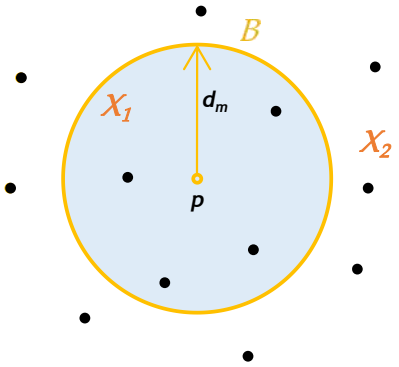
Figure 3.5a illustrates an example of ball partitioning in the Euclidean case.

Definition 3.6 (Intersection between balls)**.** Let $\mathcal{M}(\mathcal{O}; d)$ a metric space, B_1 $(\mathcal{O}, d, p_1, d_{m_1})$ and B_2 $(\mathcal{O}, d, p_2, d_{m_2})$ two closed balls. Then, there is a non-empty intersection between B_1 and B_2 if, and only if :

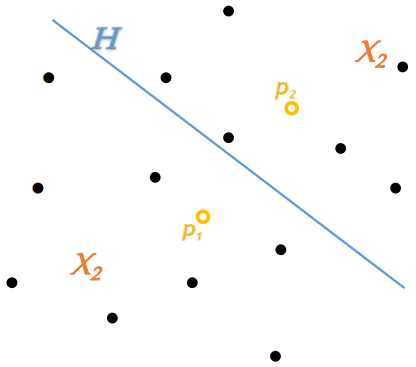
$$d(p_a, p_2) \leq r_1 + r_2 \quad (12)$$

❑ **Generalized hyperplane partitioning**

In generalized hyperplane partitioning, two distinguished objects $p_1, p_2 \in \mathcal{X}^2$ arbitrarily chosen, and the data set $\mathcal{X}$ is partitioned into subsets $\mathcal{X}_1$ and $\mathcal{X}_1$ according to the following definition 3.7.



(A) Closed Ball $\mathcal{B}(\mathcal{O}, d, p, d_m)$



(B) GH partitioning $\mathcal{H}(\mathcal{O}, d, p_1, p_2)$

Definition 3.7 (Generalized Hyperplane)**.** Let $\mathcal{M}(\mathcal{O}; d)$ a metric space, $p_1, p_2 \in \mathcal{X}^2$ two pivot points, whith $d(p_1, p_2) > 0$. Generalized hyperplane $(\mathcal{H})$ is defined as:

$$\mathcal{H}(\mathcal{O}, d, p_1, p2) = \{o \in \mathcal{X} : d(p_1, o) = d(p_2, o)\} \tag{13}$$

$(\mathcal{H})$ allows to partition the space into two sub-spaces :

$$\begin{aligned} \mathcal{X}_1 &= \{o \in \mathcal{X} : d(o, p_1) \leq d(o, p_2)\} \\ \mathcal{X}_2 &= \{o \in \mathcal{X} : d(o, p_1) > d(o, p_2)\} \end{aligned} \tag{14}$$

According to equation 14, all objects of subset $\mathcal{X}_1$ are closer to p_1 than to p_2, while the objects of subset $\mathcal{X}_2$ are closer to p_2 (see Figure 3.5b).

3.4 Approaches of indexing techniques

Several indexing techniques have been introduced to address the problems of indexing large data. This paper provides a comparative performance study of recent indexing techniques and their ability to solve large data indexing problems. In addition, these techniques are examined according to a proposed taxonomy (see Figure 3.6).

Figure 3.6 describes the classification of indexing techniques according to space. Indexing techniques can be classified into two main categories: *(i) Multidimensional Space* and *(ii) Metric Space.*

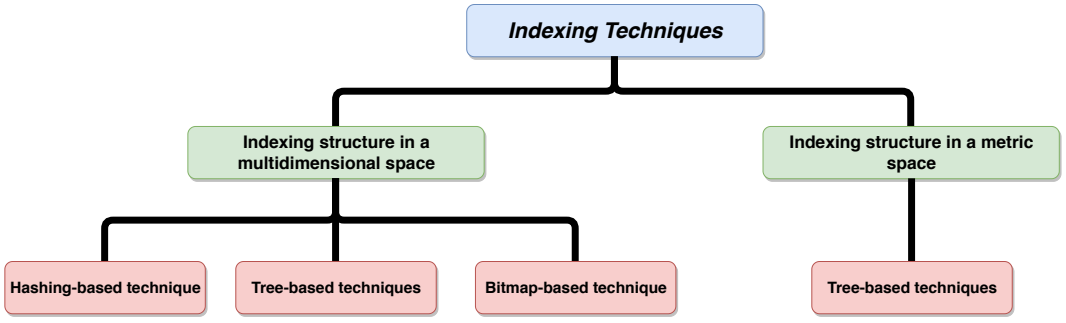


FIGURE 3.6: Global taxonomy of indexing techniques

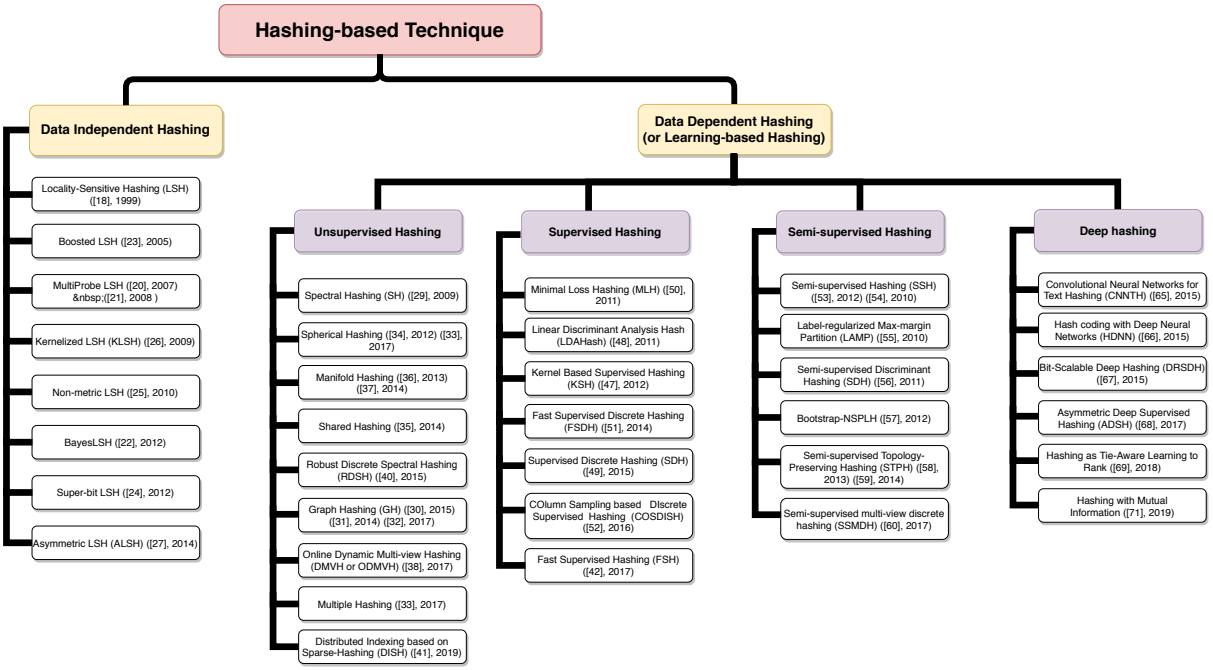


FIGURE 3.7: Taxonomy of hashing-based indexing techniques

3.4.1 Multidimensional indexing techniques

A multidimensional space is defined when the set elements are considered vectors (i.e., the data has a given number of dimensions), homogeneous or heterogeneous, whose components are totally ordered. Thus, indexing techniques in multidimensional spaces can be classified into three main types according to the type of structure used: 1) Hashing-based technique, 2) Tree-based technique, and 3) Bitmap-based technique. In the following, we review the three types of multidimensional indexing techniques.

3.4.1.1 Hashing-based technique

This is a more popular technique in the field of multidimensional data indexing due to its ability to transform a data item into a low-dimensional representation (shortcode composed of a few bits) [436]. Hashing-based indexing structures are more efficient in terms of time and storage space [122] and can detect duplicate data in a large dataset [297]. There are many methods based on the hashing technique applied to several real applications, such as computer vision, information retrieval, and analysis (e.g., images, videos, documents) [375]. According to Figure 3.7, hash-based indexing structures can be classified into two main streams: *data independent hashing* and *data dependent hashing (or learning-based hashing)*.

❑ **Data Independent Hashing**

Among the data independent hashing methods, the Locality-Sensitive Hashing (LSH) developed by Gionis et al. [157] is the most popular in literature. It allows retrieving a sufficient set of Approximate Nearest neighbours (ANNs) in high dimensional space. One of the main criteria of the LSH techniques family is the hash function which returns, with high probabilities, the same bit for close data points in the original space [439]. Since LSH's proposal, several variants

have been proposed to improve the SLH method as : MultiProbe LSH [129, 260], BayesLSH [366], Boosted LSH [374], Super-bit LSH [199], Non-metric LSH [291], Kernelized LSH (KLSH) [226] and Asymmetric LSH (ALSH) [384]. However, LSH-based techniques suffer from increasing storage costs and search time due to the long binary codes and high hash functions required when the recovery precision is improved [439]. Data-independent hash methods are well suited for small data, but they are insufficient to handle large data. Table 3.1 summarises the advantages and disadvantages of the above methods, as well as their challenges.

❑ Data Dependent Hashing

Several methods have been proposed in the hash stream to overcome the problems and limitations of data independent hash methods, depending on the data. These methods are classified into three categories according to the degree of supervision, namely: *(i) Unsupervised hashing, (ii) Supervised hashing, (iii) Semi-Supervised hashing, and (iiii) Deep hashing.*

❖ *Unsupervised Hashing*

For higher precision in the design of compact hash codes, unsupervised hashing methods aim to integrate data properties such as distributions and multiple structures [242]. Reference methods include Spectral Hashing [459], Graph Hashing [200, 252, 382], Multiple Hashing [414], Spherical Hashing [183, 414], Shared Hashing [254], manifold hashing [193, 379], etc.

Recently, a novel unsupervised online hashing method for online image retrieval was proposed by Liang et al. [473], called Online Dynamic Multi-view Hashing (DMVH or ODMVH), capable of adaptively increasing hash codes according to dynamic changes in the image. These hashing techniques also use multi-view features to achieve more efficient hashing performance. DMVH has limited performance because it is an unsupervised method and has not exploited any discriminative semantic information [257].

Yang et al. [481] developed a novel unsupervised hashing approach, named Robust Discrete Spectral Hashing (RDSH), to facilitate large-scale semantic indexing of image data. RDSH can simultaneously learn discrete binary codes and robust hash functions in a unified model. Due to the latter's difficulty, the authors included the offline process for learning binary codes and the coding functions and the online procedure for indexing images with semantic annotations. Initially, the real value representation is learned from the original space of the entities using methods such as spectral hashing (SH). Then, the real representation is transformed into binary codes through binarisation based on learning. Several experiments have been performed on various real-world image datasets to demonstrate its effectiveness in large-scale semantic indexing approaches. Compared to locality-sensitive hashing, spectral hashing generates a very compact hash code, but it is not appropriate for a large and dynamic database. A new Distributed Indexing method based on Sparse-Hashing (DISH) in cloud computing was developed by André et al. [289] to address the difficulties associated with distributing an index of high-dimensional feature vectors to multiple index nodes and search for large-scale distributed images. DISH allows documents and queries to be distributed in a balanced and redundant way between nodes. Table 3.2 provides a comparison of the technique discussed above.

❖ *Supervised hashing*

TABLE 3.1: Summary of advantage and disadvantage of data independent hashing techniques

Proposition	Ref	Advantages	Disadvantages and Challenges	
LSH	[157]	• Returns with high probabilities the same bit for nearby data points in the original space by storing similar data in the same bucket	• High storage cost • High search time • Not sufficient to processes high dimensional data	• Unsuitable to process large data
MultiProbe LSH	[129, 260]	• Reduce the number of hash table, therefore, reduce space and time compared to LSH method	• Insufficient number of neighbourhood candidates to respond to KNN's requests	
Kernelized LSH	[226]	• Search for approximate similarity in sub-linear time • No data distribution or data entry assumptions are required	• High memory consumption	
			• The search for the nearest neighbour is very difficult for high dimensional data	
BayesLSH	[366]	• High quality of search results	• Less effective performance	
Super-bit LSH	[199]	• Significant error reduction • More effective for approximate nearest neighbour recovery	• Requires long hash codes and more hash tables • High-cost of space and time	
Asymmetric LSH	[384]	• Simple and easy • Efficient for maximum inner product research	• Does not support exact search	

TABLE 3.2: Summary of advantage and disadvantage of unsupervised hashing techniques

Proposition	Ref	Advantages	Disadvantages & Challenges	
Spectral Hashing	[459]	• Does not require any labelled data • Solve a difficult non-linear optimization problem with a global optimum	• The assumption of a uniform distribution of data is usually not applicable in most cases of real-world data • Cannot directly apply in the kernel space • Does not work very well for high-dimensional data	• Less efficient than a (semi-) supervised hashing technique
Spherical Hashing	[183, 414]	• Ensuring high accuracy and a highly scalable search for the nearest neighbour	• Not sufficient for high-dimensional data • Limited performance. • Requires an expensive learning process to learn the hash functions	• Unsuitable to process large data
Robust Discrete Spectral Hashing	[481]	• Robust hash functions • Very compact hash code compared to LSH	• Not appropriate for a large and dynamic database	
Graph Hashing	[200, 252, 382]	• Suitable for large-scale applications • high search precision	• Inefficient in the search of nearest neighbours • High learning costs	
Online Dynamic Multi-view Hashing	[473]	• More efficient hashing performance	• Limited performance	
Distributed Indexing based on Sparse-Hashing	[289]	• Distribution of requests in a balanced way	• High-cost time	

TABLE 3.3: Summary of advantage and disadvantage of supervised hashing techniques

Proposition	Ref	Advantages	Disadvantages & Challenges	
Minimal Loss Hashing	[311]	• Efficient and adapts well to long code lengths • Higher search precision	• Training speed very slow Difficult to optimize	• Difficulty of finding the labeling of all data in the database
Linear Discriminant Analysis Hash	[403]	• Effective compact hashing • Less memory consumption and calculation cost	• Slower because of the extraction of SIFT descriptors	• Much slower in terms of time and effort compared to unsupervised techniques
Kernel Based Supervised Hashing	[253]	• Efficient hash functions • Higher retrieval accuracy	• Not sufficient for high-dimensional descriptors	
Fast Supervised Hashing	[244]	• Suboptimal • Fast ANN search	Not use all training points due to the complexity Unsatisfactory performance in real-world applications	• Unsatisfactory performance
Fast Supervised Discrete Hashing	[165]	• Highly efficient • Very fast and high precision • Low storage cost	• Require a significant degree of effort in large-scale applications	
Supervised Discrete Hashing	[378]	• Effective binary code learning	• Expensive training time • Insufficient precision rate	• Insufficient for high-dimensional data
Column sampling based discrete supervised hashing	[208]	• Capable to use all training data points	• Inefficient binary codes	

Supervised hashing methods are based on machine learning techniques such as decision trees [244] and neural networks [470]. These methods aim to generate intelligent indexes that can predict the unknown behaviour of the data [59, 225]. The supervised hashing methods allow treating semantic similarities and the search for medical images on a large scale [242, 322]. Many representative methods have used some form of supervision to design more efficient the hash functions: Kernel-Based Supervised Hashing (KSH) [253], Linear Discriminant Analysis Hash (LDAHash) [403], Supervised Discrete Hashing (SDH) [378], Minimal Loss Hashing (MLH) [311], Fast Supervised Hashing (FSH) [244] and Fast Supervised Discrete Hashing (FSDH) [165].

Liu et al. [253] proposed a supervised hash method with kernels (KSH) in the Hamming space, where the hash codes obtained for similar data are similar hash codes (minimises similar pairs), and for different data, the hash codes received are different hash codes (maximises dissimilar pairs). Kang et al. [208] proposed a discrete supervised hashing method, called column sampling, based on discrete supervised hashing (COSDISH). COSDISH operates iteratively, and in each iteration, several columns are first sampled from the semantic similarity matrix and then the hashing code is decomposed into two parts and alternately optimise in a discrete way. Compared to FSH [244], which cannot use all training points due to time complexity, COSDISH can use all training data points. Table 3.3 compares several supervised hashing techniques.

❖ ***Semi-Supervised hashing***

Due to the complexities of the exhaustive search of data labels in the database, semi-supervised hashing methods can use hash functions capable of training on two types of data, whether labelled or unlabeled data (partially labelled). In other words, semi-supervised hashing is a combination of unsupervised and supervised hashing [322]. The semi-supervised hashing method aims to minimise the empirical error of labelled datasets and improve the binary encoding performance. Semi-supervised hashing methods can handle semantic similarity, and dissimilarity between data [438] based on non-weighted distance and simple linear mapping. Representative methods include the Semi-supervised Hashing (SSH) [437, 438], which is considered one of the most popular methods, along with Label-regularized Max-margin Partition (LAMP) [290], Semi-supervised Discriminant Hashing (SDH) [217], Bootstrap Sequential Projection Learning for Semi-supervised Nonlinear Hashing (Bootstrap-NSPLH) [464] and Semi-supervised Topology-Preserving Hashing (STPH) [501, 502]. Lately, Zhang and Zheng in [499] presented a new semi-supervised hashing named semi-supervised multi-view discrete hashing (SSMDH). SSMDH minimises the loss jointly when using relaxation on learning hashing codes on multi-view data. SSMDH reduces the loss of regression on a portion of the labelled samples, which increases the discrimination ability of the learned hash codes. Table 3.4 compares several semi-supervised hashing techniques.

❖ ***Deep hashing methods***

Several studies have used deep learning techniques, such as in image classification [188, 231] and object detection [173, 249] methods. In addition, some hashing methods available in the literature have focused on the adaptation of deep learning techniques and, in particular, deep artificial neural networks (DANS) to take advantage of deep learning, such as Convolutional Neural Networks for Text Hashing (CNNTH) [475], Simultaneous Feature Learning and Hash Coding with Deep Neural Networks [230], Bit-Scalable Deep Hashing With Regularized Similarity Learning for Image Retrieval and Person Re-Identification (DRSDH) [508], Asymmetric Deep Supervised Hashing (ADSH) [201], and Hashing as Tie-Aware Learning to Rank [180].

TABLE 3.4: Summary of advantage and disadvantage of semi-supervised hashing techniques

Proposition	Ref	Advantages	Disadvantages & Challenges
Semi-supervised Hashing	[437, 438]	• Empirical Error Minimization • Variance and independence of binary codes maximised	• Not suitable for high dimensional data • Much slower in terms of time and effort compared to unsupervised techniques
Label-regularized Max-margin Partition	[290]	• High-quality hash functions	
Semi-supervised Discriminant Hashing	[217]	• Good separation between data labelled in different classes	• Impractical for high-dimensional data
Bootstrap-NSPLH	[464]	• Balanced partitioning of data points • Higher performance	• Expensive training time • Require storage space and a large amount of computation
Semi-supervised multi-view discrete hashing	[499]	• Minimizes the loss jointly on multi-view features when using relaxation on learning hashing codes • Increases the discrimination ability of the learned hash codes	• Not suitable for high dimensional data

Due to the automatic learning ability of the deep learning methods, deep hashing methods have shown better performance than traditional hashing methods [241]. Hash methods that adapt to in-depth learning can be based on unsupervised or supervised learning, but most of these methods are supervised, with supervised information given with triplet labels [81]. Jiang and Li [201] proposed Asymmetric Deep Supervised Hashing (ADSH) for large-scale nearest neighbour search. ADSH learns a deep hash function only for query points, while the hash codes for database points are directly learned to reduce the training time complexity. Table 3.5 below shows a comparison of some key advantages and drawbacks of deep hashing techniques.

The most significant difference between unsupervised hashing and (semi-) supervised hashing is the availability of label information for learning hash functions [440]. Compared to unsupervised hash methods, supervised methods are much slower in terms of time and effort due to the overload of the training process and the absence of label information. Thus, the unsupervised techniques are potentially valuable for practical applications as they do not require any labelled information [102, 238]. On the other hand, supervised techniques consider the advantage of explicit semantic labels of the data, which provides higher efficiency than unsupervised hashing techniques [122].

In general, to achieve satisfactory performance with data-independent methods, many hash tables or long hash codes are required, making them less effective in practice than data-dependent methods. For data-dependent, hashing methods (unsupervised, supervised and semi-supervised hashing) are needed for new solutions to address optimisation to learn hash functions and hash codes.

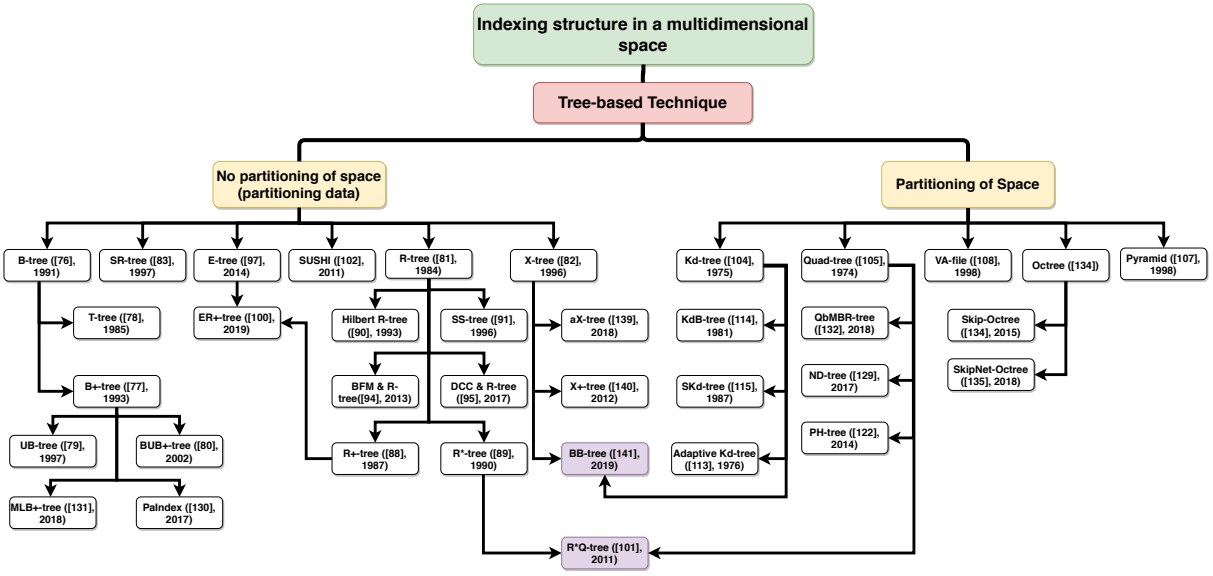


FIGURE 3.8: Taxonomy of tree-based indexing techniques

3.4.1.2 Tree-based technique

Multidimensional data has several dimensions. In metric spaces, this notion disappears; not only does it disappear because the object is only considered as a whole and not as a set of components, but also because some objects are naturally without any perceptible dimension. This is the case of a sequence of characters, a set of elements of any description, a graph, etc.

TABLE 3.5: Summary of advantage and disadvantage of deep hashing techniques

Proposition	Ref	Advantages	Disadvantages & Challenges	
Convolutional Neural Networks for Text Hashing	[475]	• Better performance than traditional hashing methods	• Unsuitable for all real-world domain databases	• Performance decreases as the dimensionality of the data increases
			• Not sufficient to processes high dimensional data	
Hash coding with Deep Neural Net	[230]	• Better performance • Good search precision rate	• Demand pairwise similarity labels • Need a more complex configuration	
Bit-Scalable Deep Hashing	[508]	• Better performance than traditional hashing methods	• Required labelled data and considerable human efforts	
Asymmetric Deep Supervised Hashing	[201]	• Reduce the complexity of training time • High search precision rate	• Learns the hash function only for query points • Higher complexity	

This section presents some tree indexing techniques. Several other authors have proposed reference books or syntheses on the subject. Also, some authors have considered multidimensional indexing techniques as unsupervised classification methods. It is important to note that in a classification, the classes are not of the same cardinal and that, in a hierarchical classification, not all leaf classes are located at the same depth. Indexing techniques can be classified according to two main approaches:

❑ ***No partitioning of space (partitioning data)***

The primary idea of data partitioning consists in creating data packets or clusters, also called "inclusion forms". In the literature, there are three data-partitioning methods: those whose minimum bounding regions (MBR) are hyper-cubes, those whose MBRs are hyper-spheres and those whose MBRs are hyper-plane [453]. The main representative techniques of this approach include the B-tree [399] (and its and its variants: B+-tree [398], B*-tree, T-tree [233], UB-tree [46], BUB-tree [144], etc.), R-tree [170], the X-tree [56] and the SR-tree [210].

R-tree is a hierarchical data structure based on B^+-tree, where it is used to index Spatio-temporal data of n-dimensions. R-tree generates several small minimum bounding rectangles (MBR) [170] to reduce dead spaces. R-tree is a balanced [5] and dynamic structure [170] that is very efficient for range requests [452]. The disadvantages of the R-tree structure reside in the increase in space, time, and complexity of the calculations because of overlapping multiple MBR regions [209]. Because of the overlapping, R-tree is inefficient for point location queries, which can degrade the performance of the search process [269].

Several extensions have been proposed based on the R-tree structure to address the weaknesses of this structure mentioned above. Among these extensions, it can be noted: R^+-tree [371], R^*-tree [47], Hilbert R-tree [207] and SS-tree [461].

X-tree (eXtended node-tree) [56] is an R-tree based structure developed to prevent overlap between MBRs through the newly proposed node type. These nodes are extended nodes of the variable size called Super-nodes (eXtended node). Due to this type of node, X-tree supports large data indexing with less overlap and less performance reduction compared to the R-tree structure. X-tree is a hybrid index that consists of a hierarchical part (tree) and a linear part (list). X-tree is a variable structure where size and complexity are difficult to calculate because of their sensitivity to size, distribution of data [63]. In addition, X-tree consumes a lot of memory space for storage, and its performance is limited to the data dimension.

R-tree nodes and their variants reduce the number of partitions that occur in the R-tree construction and increase the spatial utilisation of the R-tree to solve the problem of overlapping, which influences the construction performance and requests efficiency. Yang et al. [479] proposed a new lazy splitting strategy to optimise the R-tree generation process. Bloom Filter Matrix (BFM) is a multidimensional data indexing structure developed by Wang et al. [451] to solve the problem of decreasing index performance for high-dimension data. BFM uses a multi-dimensional matrix based on the Cartesian product of bloom filters, where each filter represents an attribute of the original data. Although the BFM structure demonstrates a multi-attribute data indexing speed and searches accuracy, it suffers from a very high space consumption [449], making it inadequate for IoT applications where data is massive. For an efficient R-tree index, Wang et al. [442] proposed a new retrieval method, called the Dynamic Clustering Center (DCC) method, which allows choosing the optimal cluster centre according to the distance indicator R during the construction of the R-tree spatial index. This technique aims to make the R-tree structure more compact, reduce multipath searches and improve search efficiency.

TABLE 3.6: Analysis of multidimensional indexing techniques based on data partitioning

Proposition	Ref	Dataset type	Data dimension	Indexing Nature	Complexity (BigO)	
					Insertion and deletion	Search
B-tree	[399]	Temporal	One-dimensional	Dynamic	$O(n\log(n))$	$O(n\log(n))$
B+-tree	[398]				$O(n\log(n))$)	$O(n\log(n))$
B*-tree	[233]				$O(n\log(n))$	$O(n\log(n))$
T-tree	[233]				$O(2n\log(n))$	$O(n\log(n))$
UB-tree	[46]	Spatio-temporal data	Multi-dimensional		$O(n\log(n))$	$O(n\log(n))$)
PaIndex	[509]				$O(n\log(n))$	$O(n\log(n))$
MLB+-tree	[443]	Seismic data			$O(n\log(n))$	$O(n\log(n))$
SR-tree	[210]	Image feature vectors			$O(n\log_3(n))$	$O(n\log_3(n))$
E-tree	[504]	Spatial			$<O(n\log(n))$	Not estimated
ER+-tree	[43]	OpinRank Review			Not estimated	Not estimated
SUSHI	[168]11	Colour histogram and Synthetic data			$O(n^2\log(n))$	Not estimated
R-tree	[170]	Geographical and Multi-media			$O(dn\log(n))$	$O(n\log(n))$
R+-tree	[371]				$O(n\log(n))$	$O(n\log(n))$
R*-tree	[47]				$O(n\log(n))$+ *Re-insertion complexity*	$O(n\log(n))$
Hilbert R-tree	[207]	Spatial			$O(\log(n) + M\log(n))$	$O(n\log(n))$
SS-tree	[461]	Multi-media data			$O(n\log(n))$ + *Re-insertion complexity*	$O(n\log(n))$
BFM & R-tree	[451]	Not mentioned			Not estimated	Not estimated
DCC & R-tree	[442]	Medical data			$O(nkt)$	$O(n\log(n))$
X-tree	[56]	Spatial data and Synthetic data			Not estimated	Not estimated
aX-tree	[359]	Spatial data			Not estimated	Not estimated
X+-tree	[127]	Spatial data			Not estimated	Not estimated
R*Q-tree	[202]	Special data			$O((kndt)(n\log(n)))$	Not estimated
BB-tree	[394]	Synthetic data, Sensor data and Genomic			Not estimated	$O(h\log(k)+b_{max}m)$ for exact-match queries

TABLE 3.7: Summary of advantage and disadvantage of multidimensional indexing techniques based on data partitioning

Proposition	Ref	Advantages	Disadvantages & Challenges	
B-tree	[399]	• Simple structure • Balanced in insertion and deletion • Efficient for k-nn and range search	• Consumes a lot of computing resources • Requires large storage space • Costly maintenance	• Support only one-dimensional data
B+-tree	[398]	• Storage at leaf nodes • Storage cost reduced compared to B-tree	• High complexity • Wasted storage space • Non-optimal node splitting	• Requires a considerable amount of computing resources
B*-tree	[233]	• Reduction of node splitting • Less storage space compared to B-tree and B+-tree	• High complexity	• Limited performance
T-tree	[233]	• Balanced structure • More efficient memory management, search and update performance than B+-tree	• Requires a considerable amount of space • Inefficient search • The problem of balance is still unresolved	• Degradation on large scale
UB-tree	[46]	• Efficient processing of multidimensional requests	• Unsatisfactory for queries covering dead spaces • Not suitable for large data	• Degradation on large scale
PaIndex	[509]	• Effective and efficient update and query performance • Structure supports parallel insertions and queries		
MLB+-tree	[443]	• Higher performance on multi-dimensional range queries	• High complexity • Sub-optimal partitioning • Irregular and unpredictable structure	
SR-tree	[210]	• Simple construction • Refinement : (intersection S ˆR) • Reduced overlap rate	• Complexity of shapes • Costly insertion and search algorithm	
E-tree	[504]	• Reduce time from linear to sublinear complexity	• High storage space • Costly maintenance • K-nn research is not evaluated • Degradation on large scale	
ER+-tree	[43]	• Reduce computation time • High quality of search results • More efficient structure		
R-tree	[170]	• Creation of filter cells REM • MBR allows you to refine your search • Balanced hierarchical breakdown • Constraint of minimum coverage	• Overlap of REMs • Not effective for point queries • Require high space and time as well as computational complexities	• Degradation of the performance on large scale
R+-tree	[371]	• Reduced overlap rate	• Redundancy of objects in nodes • Clipping technique not optimised • More complex construction and maintenance	
R*-tree	[47]	• More efficient variant than the R-tree • Reduced overlap rate • Efficient use of space	• Complexity of the re-insertion algorithm and the split of nodes	

Proposition	Ref	Advantages	Disadvantages & Challenges	
Hilbert R-tree	[207]	• Good performance results for both searches and updates	Performance deteriorates for larger data	
SS-tree	[461]	• Outperforming the R-tree • Calculate the nearest and approximately nearest neighbours efficiently	• High overlap in high-dimension space	
BFM & R-tree	[451]	• Solve the problem of decreasing index performance for high-dimensional data	• High space consumption	
DCC & R-tree	[442]	• Enhance R-tree's search efficiency • Reduce multipath searches	• Require high space and computational complexities	
X-tree	[56]	• Overlap control (overlap-free) • No degeneration of the index • Reduced overlap rate	• Complexity of the max limit • Consumes a lot of memory space • Performance is limited with the data dimension	• Cannot function properly in higher-dimensional data
aX-tree	[359]	• Reduce the amount of empty space • Reduced overlap rate • Fast loading and better partitioning of space	• Supports only static data • Require more calculation	
X+-tree	[127]	• Reduces the complexity of linear scanning of supernodes compared to X-tree	• Suffers from data redundancy and replication problems	
R*Q-tree	[202]	• Improve space utilisation • Reduce node overlap and the number of splits	• High complexity • Not suitable for the situation of frequent updates	
BB-tree	[394]	• Quasi-balanced structure • Better performance compared to R*-tree, Kd-tree, PH-tree, and VA-file	• Not support the k-nn search	

The requirement to classify data flow records such as web traffic flow monitoring, spam detection and intrusion detection is addressed in [504]. A new E-tree indexing structure with a time complexity less than $O(log_n)$ was proposed by Zhang et al. to organise all base classifiers in an ensemble for fast prediction. E-tree used a balanced height structure like an R-tree to reduce the expected prediction time from linear complexity to sublinear complexity. On the other hand, E-tree is automatically updated by the repeated aggregation of new classifiers and eliminating relevant or obsolete ones. It, therefore, adapts well to discover new trends and patterns and undifferentiated data flows [406, 504]. E-tree requires high storage space, and maintenance [296] despite the results of the analysis showing the effectiveness of this approach. ER$^+$-tree is a new multidimensional data indexing.

Balasubramanian in [43] proposed the structure on the cloud-computing infrastructure. This structure is a hybrid tree structure that combines the benefits of E-tree [504] and R$^+$-tree [371]. This structure's main objective is to reduce computation time and improve the similarity search quality in a cloud-computing environment [43]. The idea of combining these two structures is to create a more efficient structure in terms of balancing and similarity research. The E-tree structure is used to partition the data flow to reduce overload, while the R$^+$-tree structure is used to reduce search time and improve the similarity search quality through its Minimum Boundary Rectangle (MBR).

In [202], Jin and Song introduced a tree indexing structure based on R*Q-tree. This approach improves query performance and reduces indexing costs. It is based on the k-means clustering algorithm to reorganise nodes between neighbouring nodes in the tree. Also, a new indexing method (SUSHI) was proposed by Günnemann et al. [168]. This method is based on subspace clustering for indexing high dimensional objects, where the construction of the index tree is done recursively. The nodes of each level represent the groups resulting from the subspace clustering method. Wang et al. [450] presented a new approach based on searching for the nearest neighbourhood to accelerate corresponding matching faces for large-scale facial recognition systems. This method uses the k-means algorithm for clustering data and the Kd-tree structure for cluster storage. However, this technique presents, in addition to all the advantages, a problem linked to the complexity of the closing forms, which leads to an increase in the costs of insertion and search operations. Table 3.6 analyses multidimensional indexing techniques based on data partitioning, taking into account dataset type, data dimension, indexing nature, and complexity as comparison metrics, and Table 3.7 shows the advantages and disadvantages of these techniques, as well as their challenges.

❑ *Partitioning of space*

In this category, indexing techniques are based on space partitioning into sub-spaces (or cells), where each sub-space contains a subset of data. Unlike indexing techniques based on data partitioning, this type of partitioning eliminates region intersections. Many existing approaches are proposed in the literature. Reference techniques include for example: Kd-tree [53], Quadtree [146, 358], Pyramid [55] and VA-file [454].

Kd-tree (K-dimensional tree) is a binary tree structure for indexing multidimensional data based on partitioning space to k dimension using hyper-planes [53]. The main disadvantage of the Kd-tree is that it is unbalanced because the hyper-plane of space division does not divide the planes in a better position. The latter creates overlaps between neighbouring regions, which increases the cost of I/O operations [15, 54]. The performance of the Kd-tree structure to meet range requests or Knn requests is limited by data dimensions, where, as the size of the data

increases, most tree data is traversed [57, 318]. Several extensions have been proposed to address the challenges of the Kd-tree structure, the best known are: Adaptive Kd-tree [148], KdB-tree [352] and SKd-tree [317]. Like the Kd-tree, the Quad-tree [146] is the simplest multidimensional index structure, mainly used to partition a two-dimensional space by recursively dividing it into quadrants, and it includes several parts index space (each node has four leaf nodes). The Quad-tree is also not balanced because it does not choose the best division of space (horizontal or vertical) as Kd-tree. In addition, Quad-tree does not take into account the spatial distribution of data during the space partitioning phase [433].

The pyramid tree is also a multidimensional data indexing structure. The pyramid tree is based on the partitioning of the data space into 2D pyramids. Each of them is cut in a parallel slice at the pyramid's base forming the data ranges [55, 64, 327, 492]. The pyramid tree suffers from the degradation of its performance with the increase in the size of the data because the number of pyramids is insufficient to discriminate the points of high dimension. Also, Pyramid-tree creates non-discriminatory indices because the data that is located in the same pyramid slice has the same index value [27, 506].

Recently, Zäschke et al. [497] proposed the structure PATRICIA-Hypercube-tree (PH-tree) based on the binary representation of data objects as a bit string [152] and the Quadtree structure [146], which uses hypercube for space partitioning in all dimensions at each node in the tree [264]. This partitioning allowed navigating more efficiently to the sub-node and stored entries more efficient compared to the binary trees [497]. Other improvements have been proposed to improve the efficiency of the PH-tree structure [11, 428, 496]. In [11], Favre Bully added new additional functions for data pre-processing and in [428], Bogdan Aurel proposed a new distributed architecture of the PH tree for parallel processing and cluster computing. However, consistency issues and the support of ACID properties (atomicity, consistency, isolation and durability) of transactions are not investigated [234]. Furthermore, Costa et al. [114] proposed the ND-tree structure (Norm Diagonal Tree) to create a multidimensional indexing structure for high-dimensional data. It is based on a new data dimension reduction technique that uses the dual metric system and the Euclidean standard and distance to support high-dimensional data (¿100 dimensions) such as multimedia data. This technique reduces data dimension to two dimensions (2D) where they are indexed in the Quadtree tree, which is considered a better dynamic indexing structure for two-dimensional data. In this approach, the reduction method applied to indexed data inevitably causes a loss of information on the original data, which reduces the search's precision.

In the field of vehicle Internet (IoV), traffic management applications require efficient processing of requests with consideration of the massive trajectory data collected by the vehicle's tracking process. For this purpose, Zhang et al. [509] proposed an online index system for vehicle trajectory data called PaIndex. The structure of the proposed index is based on the multi-level partitioning of the space. At first, space is partitioned into regular grid cells in which the spatial domain of longitude and latitude is uniformly divided, then each cell's data is indexed in a hierarchical structure as B^+-tree. This partitioning allows parallelising the insertion operations and the search requests to reduce the time and cost. Seismic data processing applications use the requests of the multidimensional range, and to accelerate the processing of these types of requests, Wang et al. [443] proposed an extension of the B^+-tree called MLB^+-tree index (Multi-level B^+-tree). MLB^+-tree is organised in several levels, where each level contains several independent B^+-tree trees that allow the insertion and request to be performed in parallel after the top level. B^+-tree faces problems of complexity, loss of space and consumption of many computational resources in massive data due to sub-optimal partitioning of nodes.

Jo et al. [197, 204] proposed the QbMBR-tree (Quadrant based Minimum Bounding Rectangle) structure for processing large scale spatial data in HBase systems for efficient processing reduce storage space and false positives in spatial query processing. The structure proposed in this work partitions the space recursively into quadrants, and for each quadrant, an MBR is created to provide secondary indexes stored in the HBase table. The recursive partitioning is terminated until the number of objects in MBRs is less than the partition threshold. Skip-octree is a new multidimensional data index in a cloud environment proposed by Dong et al. [131]. Skip-octree is based on a two-level architecture that adapts the skip-list to accelerate the search process, and an octree structure is used in each server to store and hierarchically index multidimensional data. A new indexing technique was proposed by Malhotra et al. [268] called SkipNet-Octree based on the combination of SkipNet [176] and compressed Octree [131] to index and process queries on multidimensional data in Cloud Computing. SkipNet-Octree is a two-layer structure, where the top layer represents a global index created through the SkipNet structure that contains metadata for local index nodes and the Octree index technique used to create a local index [268]. The experiments show that the SkipNet-Octree technique works better than traditional Skiplist and Octree for complex queries.

A new hybrid multidimensional data indexing structure was presented in [408]. The structure is based on the concepts of Grid, Pyramid, and Height to partition space and design the key to access data effectively. In this structure, space is partitioned into grids, and each subspace (grid cell) is identified by pyramids, and heights [143, 408]. The main objective of this hybrid structure is to create a structure that supports floating-point numbers and reduces the number of I/Os to ensure high system throughput and more efficient execution of range requests [408].

Recently, Samson et al. [359] proposed a new static spatial data indexing structure called aX-tree (Packing X-tree) to avoid performance degradation for high-dimensional databases encountered by the X-tree [56] structure and their variances (X^+-tree [127], VA-File [454] etc.). aX-tree uses the Bulk-Loading technique to reduce empty space, fast loading and better partitioning of space based on MBR. With this technique, aX-tree has overcome the over-expansion of the super-node where it became a structure characterised by: i) minimum tree height ii) high directory node quality iii) minimum overlap and iv) reduced area of the MBR and most importantly, maximised space efficiency [359].

Sprenger et al. in [394, 396] introduced BB-tree. It is a new multidimensional index structure that combines the Kd-tree [53] and X-tree [56] structures. BB-tree is a quasi-balanced tree, supports complete- and partial-match range queries, exact-match queries, and dynamic updates. The authors created this structure based on recursive partitioning of the space into k partitions, as for Kd-tree. BB-tree is based on elastic bubble buckets in the leaf nodes of the tree-like the X-tree. These buckets store data (subset) to balance the structure. The leaf nodes (or regular BB) has a limited capacity ($b - max$). The latter is transformed into Super-Nodes (super BB) similar to X-tree structure in case of saturation; the data from these nodes is scanned linearly. According to the results of the experiments [86], BB-tree shows a better efficiency for range queries compared to R*-tree [47], Kd-tree, PH-tree [497], and VA-file [454] but Knn similarity search queries are not taken into account in this structure. Tables 3.8 and 3.9 present respectively an analytical and comparative study of multidimensional indexing techniques based on space partitioning.

3.4.1.3 Bitmap-based technique

Bitmap index (also known as BitArray or vector-based index) is an efficient indexing structure for search and retrieval of large databases and data warehouses (DW) with less complexity and is very efficient when attributes have a low number of distinct values. This technique is used by several popular commercial systems such as Oracle [29, 30] and SybaseIQ [263, 316]. Bitmap index technique is based on the representation of the existence or absence of a specific property by a sequence of bits where each bit (0/1) represents the value of an attribute for a given tuple such that the bit sequence has a 1 in position i if the i^{th} data element meets the property, and 0 otherwise [88, 469]. Bitmap index uses logical operations, such as AND, OR, NOT and XOR, to respond and accelerate responses to complex queries [466].

Traditional bitmaps are suffering from a problem of space over-consumption, especially for highly cardinal data. To address this challenge and for faster retrieval, compressed bitmap indexes are recommended. As a consequence, many efficient bitmap compression algorithms have been developed, including: BBC (Byte-aligned Bitmap Compression) [29], WAH (*Word-Aligned Hybrid*) [466, 467], PLWAH (*Position ListWAH*) [121], EWAH (*Enhanced Word-Aligned Hybrid*) [235], CONCISE (*Compressed N Composable Integer Set*) [110], VALWAH (*Variable-Aligned Length WAH*) [171], SECOMPAX (*Scope-Extended COMPressed Adaptive indeX*) [460], SBH (*Super Byte-aligned Hybrid*) [218], Roaring [87], SPLWAH (*PLWAH algorithm for Sorted data*) [92], BAH (*Byte Aligned Hybrid compression coding*) [237], cSHB (*Compressed Spatial Hierarchical Bitmap*) [294] and CODIS (*COmpressing DIrty Snippet*) [511].

Through these compression algorithms with logical operations, the execution time is reduced compared to the basic bitmap index without compression, which is an essential property of bitmap indexing [211].

Recently, Chenxing et al. [237] proposed a new compression algorithm more similar to the WAH algorithm named BAH (Byte Aligned Hybrid compression coding), whose objective is to improve the performance in terms of space and the efficiency of the requests. BAH uses simple rules for raw bitmap encoding compared to other WAH variants that use a more complicated codebook. BAH uses SIMD operations to accelerate the efficiency of the AND operation on multiple compressed bitmaps. Another compression algorithm has been proposed based on the WAH algorithm called CODIS, proposed by Wenxun et al. citezheng2017codis. The basic idea of CODIS is to reduce space through the representation of the bit string in the bitmap index with fewer bits without influencing the efficiency of the index. The results obtained during the experimentation demonstrate that this technique is more efficient than the other algorithms in the literature, including WAH [467], COMPAX (COMPressed Adaptive indeX) [149], and PLWAH [92].

The bitmap index method is a very efficient technique for answering complex queries read-only systems and for data that is not frequently updated as a data warehouse, but it is less efficient in other cases (i.e., for data frequently updated). This problem is caused by the compression process where the latter is used to reduce the storage space (as we said before), but at each update operation, it is necessary to decode and encode the bitmap, and this operation is costly [36]. Manos et al. [36] proposed a new bitmap index named UpBit (Updatable Bitmap) to overcome this problem. This index offers efficient updates without affecting read performance. The UpBit index adds an additional update vector for each bitmap vector in which update processes will be performed on the update vector, where the latter stores updates corresponding to its value bitmap only. Bitmap minimises the cost of decoding and improves navigation through the use of closing pointers on bit vectors. Chigullapally et al. [397] proposed an

TABLE 3.8: Analysis of multidimensional indexing techniques based on space partitioning

Proposition	Ref	Dataset type	Data dimension	Indexing Nature	Complexity (BigO)	
					Insertion and deletion	Search
Kd-tree	[53]	Geo-graphical	Multidimensional	Dynamic	$O(dnlog(n))$	$O(n\log(n))$
Adaptive Kd-tree	[148]	Files			$O(dn\log(n))$	$O(n\log(n))$
KdB-tree	[352]	Floating point numbers			$O(n\log(n))$	$O(n(\frac{k-1}{k}))$
SKd-tree	[317]	Spatial			Not estimated	Not estimated
Quad-tree	[146, 358]	Spatial			$O((d+1)n\log(n))$	$O(n\log(n))$
PH-tree	[497]	Synthetic			$O(n\log(n))$	$O(n\log(n))$
ND-tree	[114]	Synthetic			Not estimated	Not estimated
QbMBR-tree	[197, 204]	Synthetic, spatial			Not estimated	Not estimated
VA-file	[454]	Synthetic data and images			Not estimated	Not estimated
Octree	[131]	Spatial			$O((d+1)n\log(n))$	$O(n\log_8(n))$
Pyramid	[55]	Synthetic data			Not estimated	Not estimated

TABLE 3.9: Summary of advantage and disadvantage of multidimensional indexing techniques based on space partitioning

Proposition	Ref	Advantages	Disadvantages & Challenges	
Kd-tree	[53]	• Balanced hierarchical split • Simple implementation	• Costly and arbitrary • Low use of allocated space • Performance limited by data dimension	• Degradation on large scale
Adaptive Kd-tree	[148]	• Lower-cost k-nn research • Storage at leaf nodes	• Not appropriate for the situation of frequent insertion and deletion	
KdB-tree	[352]	• Height-balanced structure • Efficient search for point queries	• Supports only point data • Cannot guarantee minimum storage utilization • Insufficient research performance	
SKd-tree	[317]	• Suitable for non-zero size spatial objects • Ensures good storage	• Slow performance even in high dimensional spaces	
Quad-tree	[146]	• Efficient storage and retrieval	• Not balanced structure • Does not consider the spatial distribution of the data during the partitioning phase	
PH-tree	[497]	• Faster and more efficient in terms of space efficiency, query and update performance	• Supports point and range query only • High memory consumption	
ND-tree	[114]	• Support high-dimensional data	• Loss of information on the original data • Search performance limited by data dimensions	
QbMBR-tree	[204]	• Reduce the false positives in spatial query • Reduces the storage space • Reduce query execution times	• Overlap of MBRs	
VA-file	[454]	• Simple implementation • Sequential search improved	• Large dimension heavy coding • Degradation on large scale	
Octree	[131]	• Better spatial management and k-nn search	• Support only 3-dimensional data	
Pyramid	[55]	• Degradation on a large scale • Linear increase of cells	• Poor request processing k-nn • Degradation on large scale	

extension of this structure, where the authors parallelised the merging of bit vectors to improve the performance of the UpBit index.

3.4.2 Metric indexing techniques

This part introduces a study of a variant of a metric tree data structure for indexing and querying such data.

3.4.2.1 Partitioning of space

In the literature, two partitioning techniques have been developed: the first technique is based on hyper-sphere (or ball) partitioning as in: VP-tree [486], mVP-tree [69] and MM-tree [328], etc. while the second technique is based on hyper-plane partitioning as in: GH-tree [424], GNAT-tree [72] and EGNAT-tree [321], etc.

❑ **Hyper-sphere (or ball) partitioning**

VP-tree [486] is a hierarchical indexing structure such as Kd-tree, developed to improve the search for similarity in a metric space. VP-tree is a technique based on the partitioning of the space through the balls according to distance. VP-tree uses the median distance between the vantage point (choose randomly) and the points of the space to partition the space into two balanced disjoint sub-spaces. The disadvantage of the VP-tree is the highest cost in terms of calculated distance and time, especially in the data space whith large dimensions where the number of branches searched for is high [512]. The mVP-tree is proposed to address the problem of reduced performance in the search for similarity of the VP tree in high-dimensional metric spaces. mVP-tree (multiple Vantage Points tree) [69] is an extension of the VP-tree idea that uses several vantage points instead of one. The major advantage of mVP-tree over VP-tree is that it also uses pre-computed distances (at the construction step) to improve search speed and reduce the number of distance calculations and the time required to execute queries. The experiments presented in [486] show that mVP-tree improves the VP-tree slightly better but not in all cases, while more remarkable improvement is achieved when several pivots per node are used [94].

Cheng et al. [101] proposed the DMVP-tree structure to accelerate the recovery process of similarity images in the airport's video surveillance system. This approach improves the metric indexing structure MVP-Tree [69] using the horizontal distribution of MVP-Tree structure in several machines to overcome massive high-dimensional spatial indexing problems. The DMVP-tree structure partitions the space horizontally, where the upper area is called "mainspace" and the lower areas are called "secondary spaces". The main space is indexed in MVP-Tree stored in the master machine, and the secondary space is partitioned statically on the slave machines.

MM-tree [328] is another indexing structure for metric space that also uses the principle of recursive partitioning of space through balls in two non-overlapping regions. MM-tree is an unbalanced structure due to the different sub-spaces (or regions) size due to the external region of the balls. To solve this problem, MM-tree applies an additional semi-balanced algorithm that allows re-organising the objects of the leaf nodes. According to the experiences presented by [82], MM-tree does not support high-dimensional data. Several structures have been proposed based on the MM-tree structure. All these structures are developed to address the challenges

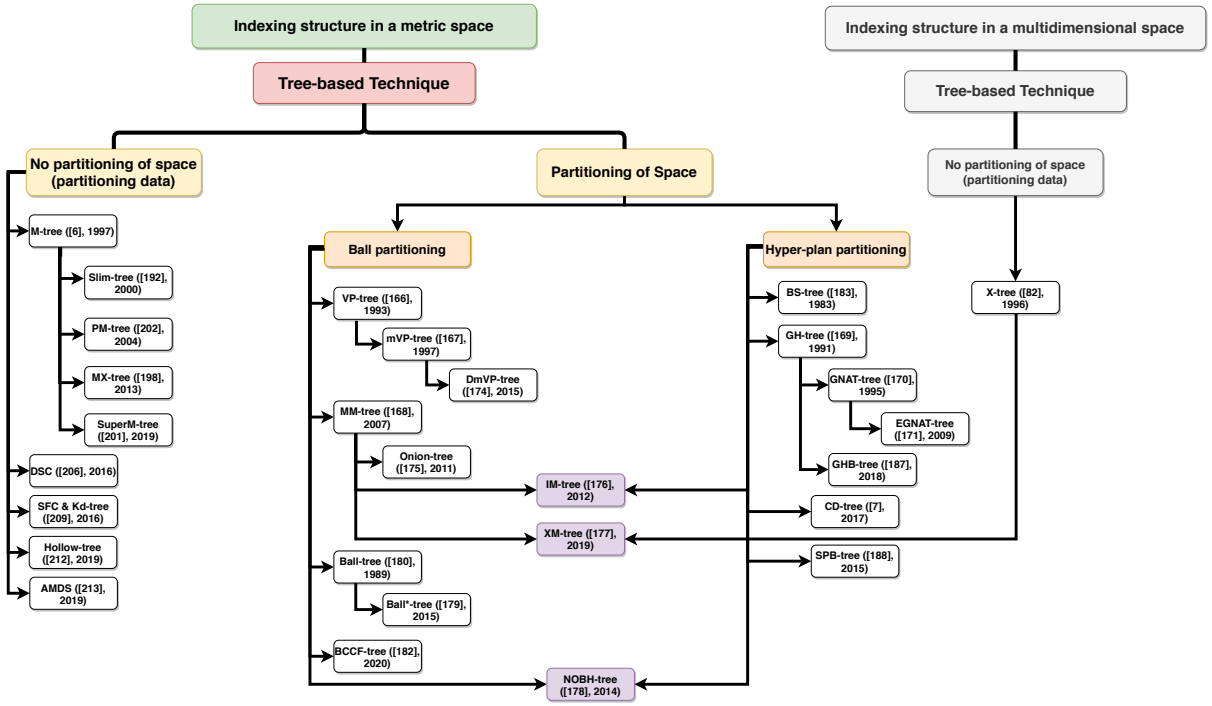


FIGURE 3.9: Taxonomy of tree-based indexing techniques in metric space

of the MM-tree structure (Onion-tree [82], IM-tree [222] and XM-tree, the extended Metric tree [221]).

Onion-tree [82] is an MM-tree improvement proposed to overcome the challenges or limitations of the MM-tree structure. The Onion-tree is very fast to respond to the similarity search requests thanks to the increase in the number of partitions of the space compared to the MM-tree, but the problem with this structure remains in the prolonged building because of the re-insertion of objects. IM-tree [222] is a proposed structure to address the issue of index degeneration posed by the fourth region of MM-tree and onion-tree. IM-tree selects the two most distant points as pivots and splits the fourth region in two using a plane. For massive data, the external region of the IM-tree balls becomes very large, which can then lead to the degeneration of the index.

XM-tree [221] is an extension of the IM-tree [222] structure that is based on the successive division of space with spheres. XM-tree is proposed to address the problem of degeneration of the index mentioned above by focusing on minimising the size of the outer regions of the balls. To achieve this goal, XM-tree creates extended regions inspired by the X-tree [56]. The extended regions make the Knn search very fast, thanks to eliminating some objects that are not necessary to calculate the relative distances of a query object.

With the same IM-tree principle, the NOBH-tree (Non-Overlapping Balls and Hyper-planes tree) [329] partitions the metric space through the hyper-planes and hyper-spheres to organise the data into non-overlapping regions as well as to reduce the number of distance calculations required to answer the questions. The NOBH tree recursively divides the space into several regions using the pivots (p_1, p_2), and separates the data such that the distance evaluation of an element X_i at p_1 and p_2 can only contain the region X_i. These regions are divided using a metric hyper-plane and two spheres, where the radius of the sphere r corresponds to the distance between p_1 and p_2. The main drawback of this technique is the complexity of the enclosing forms. This increases the cost of insertion and searches operations [221].

Ball*-tree [128] is a binary tree more balanced where each node defines a D-dimensional hyper-sphere, or a ball, that contains a subset of the points to be searched. Ball*-tree is an improvement of the original structure of Ball-tree [251, 315] proposed by Dolatshah et al. in [128]. Ball*-tree addresses the problem of data distribution and the unbalanced structure of Ball-tree by taking into account the data distribution when determining the splitting hyperplane. In Ball*-tree, the splitting hyperplane is perpendicular to the first principal component using principal component analysis (PCA). Using this splitting technique allows creating a more balanced and efficient tree structure, unlike the Ball-tree, where the splitting hyperplane is determined by the line connecting the two furthest points, creating unbalanced sub-partitions.

Tables 3.11 and 3.12 present respectively an analytical and comparative review of metric indexing techniques based on ball partitioning.

❑ Hyper-plane partitioning

The first indexing structures that are based on the partition of space through hyper-planes are the oldest structure BS-tree (Bisector tree) [206] and the structure GH-tree (Generalised Hyper-plane tree) [424] which is similar to BS-tree. GH-tree is a binary indexing structure that divides the space recursively into two sub-spaces through the hyper-plane, which is defined by the two representative points or pivots (the two farthest points as in [136], [277], and [283]) and the rest of the points are partitioned according to the distance between these pivots. The drawbacks of this structure reside in the search process where at each node, two distance operations are performed, which increases the cost of the search, as well as the selected pivots does not guarantee the best partition of space, which makes the problem of index degeneration possible. GNAT-tree (Geometric Near-neighbour Access Tree) [72] is a static indexing structure. GNAT-tree generalises the GH-tree, which uses m pivots in each internal node instead of two (i.e., GNAT-tree is an m-ary tree). Regarding EGNAT-tree [321], it is the dynamic structure of GNAT-tree.

Recently, GHB-tree (Generalised Hyper-plane Bucketed) [220] is proposed as an improvement of the GH-tree structure. The objective of the GHB-tree structure is to create a balanced indexing structure with less construction cost through the new type of node that they called a bucket. These nodes found at the leaf level have a limited capacity to store a subset of the most similar data to improve the search process. The CD-Tree, cited in [434], is a type of index based on hyper-plane partitioning. This indexing approach has proven effective for a limited number of dimensions, but remains ineffective for large dimensions. The recursive partitioning of space into two regions is the principle of this technique. Two pivots are chosen each time, and each one is associated with the closest objects. However, this technique's problem is that the geometrical shapes of the regions pose many problems in the search algorithm.

A new metric indexing structure called SPB (Space-filling curve and Pivot based B^+-tree) tree was proposed by Chen et al. [97, 98]. The method was proposed to improve the efficiency of similarity search, support a large number of complex objects, and reduce the cost in terms of storage, construction, and search (i.e., reducing CPU and I/O cost). SPB-tree uses geometric information not available in metric space by mapping objects in a metric space to data points in a vector space using well-chosen pivots to achieve these objectives. The B^+-tree with MBB (Minimum Bounding Boxes) is used to index the one-dimensional data generated by the function of dimensionality reduction SFC (Space-Filling Curve) applied to the data points of the vector space. Although the structure is very simple, but the construction steps such as space transformations and pre-treatment can make parallelism very difficult [324].

Compared between the two partitioning strategies (hyper-sphere and hyper-plane), it can be observed that the problem of node overlap is a problem that has not been effectively addressed by the techniques based on partitioning by hyper-sphere, but this problem does not exist in hyper-plane techniques (such as GH-tree and GNAT). On the other, structures based on hyperplane partitioning are more difficult to maintain their balance because of the uncontrolled insertion positions of new elements [321]. A comparative and analytical study of metric indexing techniques based on hyper-plane partitioning is presented in Tables 3.14 and 3.15, respectively.

3.4.2.2 No partitioning of space (partitioning data)

This category does not require space partitioning. We find the M-tree family essentially among the families that use this type of partitioning (partitioning data). M-tree [105] is a metric tree structure height-balanced, allowing incremental updates based on the grouping of dynamic data into balls (or hyper-spheres). M-tree stores some data in internal or inner nodes for routing purposes, and the remainder is stored in the leaf nodes. M-tree suffers from the problem of overlapping sub-spaces, which increases the number of distance calculations to answer a query [221, 513]. Several recent structures that share the same principles as the M-tree and Slim-tree [420] are among them. Slim-tree improves the M-tree structure with a new splitting technique based on the minimum spanning tree (MST). Slim-tree also reduces the cost of construction. In addition to this, it introduces a post-processing method that reduces overlap and, consequently, the cost of research. The major disadvantage of this algorithm is the ability to generate nodes with few objects and/or empty nodes, which significantly reduces the performance of the index, especially in large spaces [390, 419, 498].

The work of Murgante et al. [203] aims to avoid unsatisfactory node partitioning and reduce regional overlap in the M-tree structure [104, 106]. The authors proposed a new metric indexing structure called M^X-tree based on the original M-tree structure. M^X-tree implements the concept of super-nodes inspired by the [56] structure of the X-tree. This structure avoids the unsatisfactory division of nodes, thus reducing computation cost and extending it completely to metric space where temporal complexity is reduced to $O(n^2)$ without setting any parameter. As for the M tree, the temporal complexity reaches $O(n^3)$. The authors also add another strategy to the M^X-tree structure to improve the management of free memory space that is represented in the indexing of tree leaf objects in an internal index [340] through the vantage-point tree (VP-tree) [487]. Due to the symmetry of the metric axioms of metric space, metric indexing techniques such as M-tree and their variances cannot answer the approximate requests of subsequences or subsets. Bachmann [42] propose an improvement on the M-tree structure called SuperM-tree to create a metric indexing structure capable of responding to the approximate requests of sub-sequences or subsets such as searching for a similar partial sequence of a gene, a similar scene in a film, or a similar object in an image. To create this structure, the author introduces a new metric measurement subset space, "Metric Subset Space (M; d; v) ". It ignores the symmetry of metric axioms and adds a new relationship to object size (for more details on the demonstration of this new space, see article [42]).

The efficiency of the search in M-tree is reduced when the volume is high; thus, Pivoting M-tree (PM-tree) is proposed [388, 389] to resolve this problem. PM-tree is a hybrid structure, which combines the "local-pivoting strategies" of M-tree [105] with the "global-pivoting strategies" of LAESA [285]. Recently, Razent et al. [346] presented a new construction algorithm for the two indexing structures M-tree and PM-tree. The objective is to enhance the performance of Knn requests. The construction algorithm is based on storing data once in the tree (M-tree or PM-tree) through the deletion of promoted elements stored in the upper level of the leaf nodes

TABLE 3.11: Analysis of metric indexing techniques based on ball partitioning

Proposition	Ref	Dataset type	Data dimension	Indexing Nature	Complexity (estimation)	
					Insertion and deletion	Search
VP-tree	[486]	Images	Dynamic	Multidimensional	$O(n\log_2(n))$	$O(n^2\log(n))$
mVP-tree	[69]	Images	Static		$O(n\log_m(n))$	$O(mn\log(n))$
MM-tree	[328]	Image and Geographic coordinates	Dynamic		$O(n^2\log_4(n))$	$O(n^2\log(n))$
Onion-tree	[82]	Image, Time-series and Geographic coordinates			$O(n^2\log(n))$	$O(n^2\log(n))$
IM-tree	[222]	Image			$O(n\log(n))$	$O(c_{max}\log(n))$
XM-tree	[221]	Geographic coordinates and Image			$O(nmx\log(n))$	$O(c_{max}\log(n))$
Ball-tree	[251, 315]	Not mentioned			$O(n\log(n))$	$O(d\log(n))$
Ball*-tree	[128]	Synthetic and Point data			$O(n\log(n))$	$O(\frac{n}{d}\log(n))$
NOBH-tree	[329]	Image and Synthetic			$O(n\log_m(n))$	$O(n\log(n))$

TABLE 3.12: Summary of advantage and disadvantage of metric indexing techniques based on ball partitioning

Proposition	Ref	Advantages	Disadvantages & Challenges	
VP-tree	[486]	• Simple implementation	• Highest distance and time • Research costs increase in large dimensions	• Degradation on large scale • The problem of overlap not effectively addressed
mVP-tree	[69]	• Reduces research costs • Little affected on a large e scale	• Static structure • Support only range research	
MM-tree	[328]	• Best space partitioning • Non-overlapping regions	• Degeneration of the index (fourth region)	
Onion-tree	[82]	• Better partitioning of space	• "Reinsertion" objects (semi-balancing)	
IM-tree	[222]	• Efficient compared to MM-tree and Slim-tree	• Index degeneration in massive data	
XM-tree	[221]	• Minimise the size of the search regions • Fast k-nn search	• Requires high memory space	
Ball-tree	[315]	• Efficient brute force search in large dimensions	• Unbalanced structure • Longer build times	
Ball*-tree	[128]	• More balanced and efficient structure compared to Ball-tree	• Performance decreases as the dimensionality of the data increases	
NOBH-tree	[329]	• Non-overlapping division of the data space	• High-cost of insertion and research	

TABLE 3.14: Analysis of metric indexing techniques based on hyper-plane partitioning

Proposition	Ref	Dataset type	Data dimension	Indexing Nature	Complexity	
					Insertion and deletion	Search
BS-tree	[206]	Point data	Multidimensional	Static	$O(n \log_2(n))$	not estimated
GH-tree	[424]	Not mentioned			$O(n \log_2(n))$	$O(n \log(n))$
GNAT-tree	[72]	Image, text, Vectors			$O(nm \log_m(n))$	$O(\frac{n}{m} \log(n))$
EGNAT-tree	[321]	Words and coordinate space		Dynamic	$O(nm \log_m(n))$	$O(\frac{n}{nd} \log(n))$
GHB-tree	[220]	Geographic coordinates and Image			$O(n \log(n))$	$O(2k \log_2(k))$
CD-tree	[434]	Image			$O(n \log(n))$	$O(n \log(n))$
SPB-tree	[97, 98]	Words, Colors, DNA, Signature and Synthetic			$O(nlx + nm \log_m(n))$	not estimated

TABLE 3.15: Summary of advantage and disadvantage of metric indexing techniques based on hyper-plane partitioning

Proposition	Ref	Advantages	Disadvantages & Challenges	
BS-tree	[206]	• Fast k-nn search and orthogonal queries	• Requiring linear space	• Degradation on large scale
GH-tree	[424]	• Simple partitioning • Reduced overlap rate	• Complicated form to manipulate • Degeneration of the index • High-cost search	
GNAT-tree	[72]	• Non-overlapping • Improve the search	• Static and complicated structure • High computational costs	• More difficult to maintain index balance
EGNAT-tree	[321]	• Non-overlapping • Requires less CPU time than the GNAT-tree	• Degradation on large scale • More difficult to maintain index balance	
GHB-tree	[220]	• Balanced structure		
CD-tree	[434]	• Efficient re-construction time	• Ineffective in large dimensions • Ineffective search	• Degradation on large scale
SPB-tree	[97]	• Simple structure • Reduce the cost in terms of storage, construction and search • Effective similarity search	• Difficult to parallelise it	• More difficult to maintain index balance

during their partitioning. To achieve this idea, the authors use the aggregate nearest query to find the most efficient local pivots that will be promoted during the partitioning of internal nodes. According to the report of the experiments carried out in [346], this algorithm reduces node occupancy, reduces overlap between nodes and increases significantly the performance of search operations in terms of speed compared to the construction algorithm of the M-tree and PM-tree structure.

Navarro et al. [300] proposed the DSC (Dynamic Set of Clusters) structure, a new dynamic metric index structure that reduces memory consumption. DSC is a combination of two new structures proposed in [300]. The first structure is a hierarchical structure called DSAT (Dynamic Spatial Approximation Tree). This structure uses timestamps that indicate when elements were inserted to avoid reconstructing the structure after updates, and the pruning process of similarity queries [314]. The second structure is a variant of the LC (List of Clusters) structure called DLC (Dynamic List of Clusters). DLC is a secondary memory-based structure in which it reduces memory consumption compared to the original LC [174], where the M-tree structure is used as a partitioning technique. DSC is a structure divided into two parts. A part stored in the main memory as a DSAT structure, and the second part stored in the disk, represented by the DLC structure [300].

Through the MapReduce framework, Chanet et al. [96] proposed two partitioning techniques for joins of metric similarity to balance the load [45]. The first method selects centroids and clustering data in a one-dimensional space through the Space-Filing-Cuvre (SFC) technique. This technique allows partitioning the data in equal size thanks to the high quality of the selected centroids—the second partitioning method is based on the Kd-tree structure [53, 465], which divides the data after the pivot mapping [96].

Because of missing data generated by different application areas, indexing structures are distorted, where the latter produces a bias in response to the query. Brinis et al. [73] proposed the Hollow-tree structure to solve this problem, which enables missing data to be managed without distracting from its structure. Hollow-tree is a metric access method that uses the CFMLI (Complete First and Missing Last Insert) technique to provide a strategy for building metric indices. This strategy consists of indexing all complete data in the first steps to creating a coherent structure, then insert the elements with the missing values (with NULLS) at the nodes of the sheets. All this is achieved by the ObAD (Observed Attribute Distance) technique, which makes it possible to compare elements with missing values based on distance functions.

Yang et al. in [478] proposed an asynchronous metric distributed system (AMDS) for metric spaces to process metric similarity requests efficiently in a distributed environment. In the proposed system, the authors adopt the pivot mapping technique, which enables to divide the data uniformly into non-joint fragments and provides load balancing. To reduce computation costs in similarity research, the minimum bounding box (MBB) technique is used. The AMDS system supports large-scale similarity requests in metric spaces simultaneously through synchronous processing based on publication/subscription communication mode. Tables 3.17 and 3.18 analyse and compare metric indexing techniques based on data partitioning.

3.5 Conclusion

This chapter presents the second application context of our research, namely the indexing of big IoT data. In section 3.2, we introduced big data and its relationship to IoT data, while in the rest of the chapter, we focus on techniques for managing and indexing big data. Several authors

have already pointed out that metric space has become a popular model for overcoming the limitations of vector or multidimensional spaces in different applications. To this end, in section 3.3, we have provided the necessary elements to understand indexing and data retrieval in metric spaces as a data abstraction space. In section 3.4, we presented a comprehensive overview of the literature on indexing Big IoT data. We have presented a new indexing technique taxonomy that relieves researchers from a slow reading of related works. We also analysed, compared, and classified in-depth these IoT data indexing techniques.

TABLE 3.17: Analysis of metric indexing techniques based on data partitioning

Proposition	Ref	Dataset type	Data dimension	Indexing Nature	Complexity	
					Insertion and deletion	Search
M-tree	[105]	Synthetic data	Multidimensional	Dynamic	$O(mn\log_m(n))$	$O(mn\log(n))$
Slim-tree	[420]	Spatial, Face vectors and Text			$O(n^2\log(n))$	$O(n^2\log(n))$
Slim*-tree	[330]	Image and Spatial			$O(n^2\log(n))$	$O(\frac{n^2}{d}\log(n))$
MX-tree	[203]	Image, Text			$O(n^2\log(n))+n^2)$	$O(n^2\log(n))$
SuperM-tree	[42]	Synthetic data			not estimated	not estimated
PM-tree	[388, 389]	Synthetic data			$O(n(m+l)\log_m(n))$	$O(n^2log(n))$
DSC	[300]	Vectors, Text, and Colours			not estimated	not estimated
SFC & Kd-tree	[96]	Synthetic, Text, DNA, and Colour			$O(n\log(n))$	$O(kn\log(n))$
Hollow-tree	[73]	Synthetic			not estimated	not estimated

TABLE 3.18: Summary of advantage and disadvantage of metric indexing techniques based on data partitioning

Proposition	Ref	Advantages	Disadvantages & Challenges	
M-tree	[105]	• Balanced height structure • Reduction of distance calculations	• Problem of overlaps • High-cost search • No adapted to highly grouped data	• Degradation on large scale
Slim-tree	[420]	• Efficient compared to M-tree • Reduced overlap rate	• The overall computational complexity	
Slim*-tree	[330]	• Reduces the cost of calculation during reconstructing • Avoids the unsatisfactory division	• Reinserting objects is largely costly	
MX-tree	[203]	• Reduces the cost of calculation during reconstructing • Avoids the unsatisfactory division	• High-cost search	
SuperM-tree	[42]	• Capable of responding to approximate requests for subsequences or subsets	• Expensive construction • Evaluates only for research 1-nn	
PM-tree	[389]	• More efficient similarity search compared to M-tree	• Not support the k-nn search • Expensive construction compared to M-tree	
DSC	[300]	• Reduces memory consumption	• High amount of distance calculations	
SFC & Kd-tree	[96]	• High quality of the selected centroids • Effective partitioning • Better query performance	• Not support the k-nn search	
Hollow-tree	[73]	• Capable of managing missing data	• Lower accuracy in small data	

Part II

Proposed Collaborative Video Surveillance System for Suspicious Behavior Analysis

CHAPTER 4

DISTRIBUTED COLLABORATIVE MULTI-CAMERAS FOR TRACKING MOVING OBJECTS BASED ON INTERNET OF VIDEO THINGS -IOVT- PARADIGMS

Chapter contents

4.1 Introduction

In this chapter, we will propose an intelligent video surveillance system capable of tracking moving objects in a large-scale surveillance area, such as in smart cities. Our proposed approach's main objectives are improving tracking quality, the quality of the behavioural analysis, and reducing overhead costs (energy, bandwidth, and storage). Several issues need to be addressed to design this system for a large-scale multi-camera network and these purposes.

4.1.1 Research questions and hypotheses

RQ1. ***How to improve the quality of object tracking in a multi-camera video surveillance system?***

Several criteria can be used to determine the quality of monitoring. Among these criteria, the number of objects tracked and their tracking duration during their presence in the surveillance zone.

The system assigns an identifier (or label) to each tracked object to distinguish it from other objects during the tracking process. This identifier must be unique, regardless of the object detection area or time. Changing this identifier means that they are not the same objects. In conventional systems, there are cases where the system may give more than one identifier for the same object during tracking, and this change is known as an ID-switch error. As ID-switch error increases, the tracking time is decreased; therefore, the system's quality is decreased. Consequently, it is necessary to determine the cause or source of the problem to address it.

The ID-switch error can appear at two levels: (1) *at the camera level (low level)*. At this level, the ID-switch error is caused by object detection or association errors because of trajectory intersection or object overlap with other objects. In other words, the ID-switch error at this level results from the failure of the tracking techniques used. (2) *At the network level (high level)*. At this level, the ID-switch problem usually occurs when the tracked object leaves the camera's FoV or the zone and enters the blind/dead zones. In such a case, when the objects re-enter another camera FoV or zone, they are considered as a new object. Therefore, we can conclude that this problem results from a lack or absence of information and knowledge between cameras about the tracked objects, either in the same area/region or between different areas of the system. According to our study in chapter 2, the mechanism that allows cameras to share knowledge is coordination and

collaboration between them (see section 2.4.4). Based on this statement, the system that we will propose must be based on multi-camera collaboration.

RQ2. ***How to reduce system overhead?***

Traditional video surveillance systems require more power, large bandwidth, and extra storage space during acquiring, processing, and transmitting multimedia data in the network. Is it possible to reduce these requirements knowing that they will increase as the number of cameras increases, especially for large-scale multi-camera systems?

According to [166], conventional video surveillance systems suffer from several problems related to recording scenes containing no useful information (in our context, no object detected) and the redundant recording of the same scenes by several cameras (see Figure 4.1). These redundant and unnecessary scenes increase the amount of multimedia data collected, which requires more resources such as bandwidth, storage, and computing power to transmit, store, process, and analyse them. At this point, the question is, " *What is the cause of the redundancy?*".

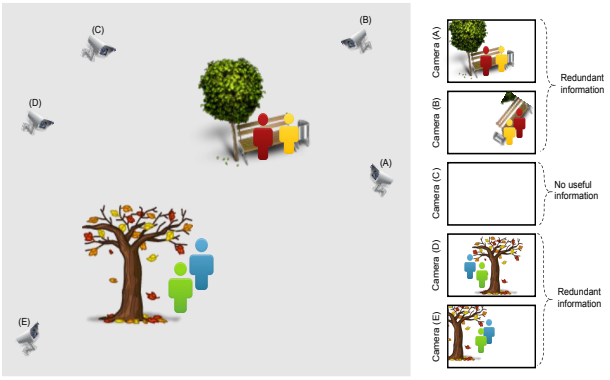


FIGURE 4.1: An illustrative multi-camera environment with overlapping FoV.

The detection of the same event by several cameras means that they have a common FoV between them. This common FoV area is known as the overlapping FoV. While the cameras will not have any knowledge about their neighbours and particularly about their overlapping neighbours, this redundant detection will undoubtedly occur. Hence, collaboration is needed to exchange knowledge of the cameras between them and coordinate and manage them if an event occurs in the overlapping area.

RQ3. ***How to improve system performance for real-time operation?***

The study in chapter 2 proves that the traditional centralised system is unsuitable for real-time tracking objects due to the many challenges (back to section 2.4.3). For this reason, the development of a decentralised system is mandatory. The principle of decentralisation is the distribution of the system load on the different nodes of the system. To achieve this, we need an infrastructure that supports this type of system. This infrastructure is delivered by the IoT and its modern computing paradigm, discussed in Sections 1.3 and 1.4 of Chapter 1. Accordingly, the integration of these paradigms into the proposed system has several issues that must be taken into account to create an efficient real-time video surveillance system:

- Is it possible to implement this system at multiple levels?
- How to partition the surveillance system load between these levels?

- To reduce network bandwidth, overall cost, and efficiency, how to select the partition and the steps to be performed?

RQ4. ***In a large-scale surveillance system, is it possible to establish a collaboration mechanism between all system cameras?***

Collaboration between cameras for tracking objects in video surveillance systems is an efficient solution to increase system performance. Despite their advantages, this latter's complexity makes it challenging to develop a coordination mechanism, particularly as the number of cameras increases. Besides, an excessive amount of communication is required, and it also increases considerably with the number of cameras, which requires a large bandwidth and more energy consumption. One of the suggested subjects to overcome this weakness and improve these systems' efficiency is the clustering or grouping of cameras. The main objectives of camera clustering are to achieve the ability to coordinate between cluster cameras instead of all system cameras in detection and processing tasks. Collaboration between cameras can significantly help avoid wasted energy by avoiding redundant event detection, processing, and sending data. Thus, it extends the network lifetime, especially in dense networks typically deployed with a high number of cameras. Nevertheless, one critical question that has not been addressed so far is, " *Which suitable cameras should be grouped?*".

The answer to this question is related to the purpose of the system. Our objective is to establish a coordination mechanism between cameras to avoid wasting energy by avoiding redundant detection, processing or data sending. Furthermore, according to question (RQ2), the redundancy is caused by the overlapping FoVs of multi-cameras, where events are detected simultaneously by these overlapping cameras. Based on this, we can assume that coordination should be applied to groups of cameras that have a high degree of overlap. Thus, as an answer to the previous question: "the group should be composed of overlapping cameras, with a maximum number of cameras in the same group".

RQ5. ***How to avoid the congestion problem in a large-scale multi-camera network?***

The grouping of cameras discussed in question (RQ4) solves the problem of communication between the cameras. But "what about the bottleneck problem?". The bottleneck problem occurs when several cameras in the system send many data to one sink (or base station). This problem leads to network congestion, which negatively influences our system, especially on real-time tasks. Combining all the information collected by the cameras inside the cluster foci clusters, called cluster head nodes (CHs), is one of the fundamental methodologies proposed to solve this problem. Unfortunately, the CH cameras consume a lot of power between transmissions to the base station and other cameras. This latter leads to unbalanced power consumption during cluster communications. For this reason, the system that we will propose should be based on a dynamic CH camera selection mechanism.

Based on the research questions, we established criteria for the system that we will propose. The system should be based on the IoVT paradigm as an infrastructure that allows us to implement a multi-camera video surveillance system on a large scale and support real-time task operations (RQ3). Object tracking-based collaborative cameras should be used to increase the tracking's quality by knowledge sharing about the objects being tracked (RQ1) and to reduce the resources consumed through coordination between them by avoiding redundant event detection (RQ2). Concerning the multi-camera network topology, all system cameras must be grouped according to the FoV overlap area to ensure that coordination mechanisms must be applied to the most

overlapping cameras to restrict the communication area and avoid network saturation (RQ4, RQ5).

The remainder of the chapter is organized as follows. Section (4.2) presents the proposed architecture of our system based on the IoVT paradigm. Section (4.3) describes the proposed multi-camera grouping strategies. Section (4.4) presents the proposed distributed and collaborative tracking system. Finally, section (5.5) concludes our chapter, including some perspectives.

4.2 Proposed architecture for IVSS in IoVT

Video surveillance has become ubiquitous due to the increasing security requirements in all areas of life. The next generation of VSS poses significant challenges in various applications, such as intelligent urban surveillance systems and smart cities. In these applications, the VSS needs to deal with the fast-growing number of surveillance nodes which impose several requirements, like high latency, high bandwidth, high energy consumption, processing power, and storage capacity (discussed in Chapter 2). To meet these requirements, the IoVT can be a promised solution. The IoVT is composed of smart cameras connected to the internet. Unlike conventional VSS, VSS-based IoVT provides multiple layers (i.e., mist, edge, fog, cloud, etc.) of communication and decision-making by capturing and analysing rich contextual and behavioural information.

This section introduces a new distributed architecture of our video surveillance system based on the IoVT computing paradigm to improve real-time object tracking quality. As shown in Figure 4.2, the proposed system is distributed over five layers. Each layer has specific tasks among the tasks of real-time tracking of moving objects. These tasks are assigned according to the task requirements and the resources available in the layer. For example, time-sensitive data are executed at the layer closest to the cameras as possible. In contrast, the other data are assigned to the farthest layer. In the rest of this section, we will describe each layer's contribution to our system.

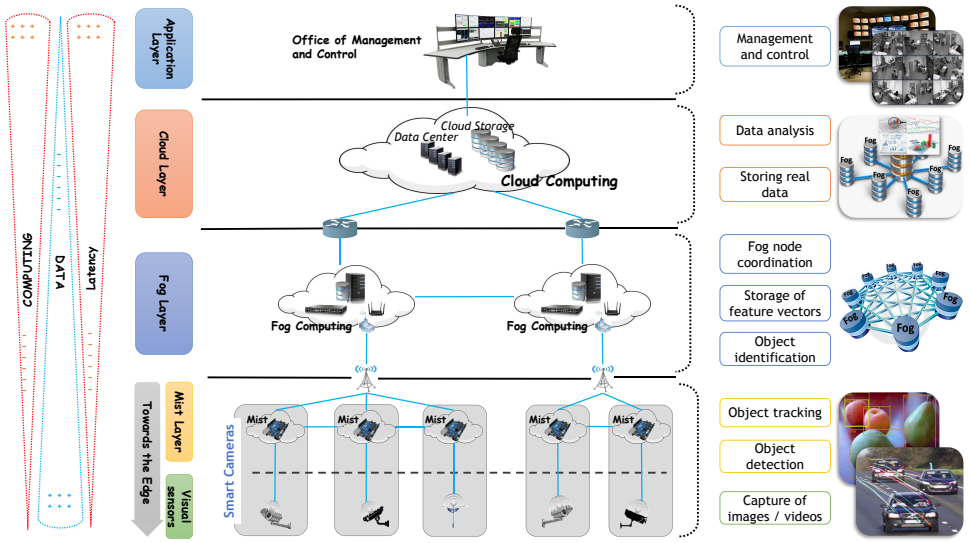


FIGURE 4.2: Proposed system architecture

The lower layer presents the connected smart cameras distributed in an intelligent environment. As we defined them in section 2.4.2 of chapter 2, smart cameras are cameras that has processing, storage, and communication capabilities. Either they may integrate as a single entity, as shown

in Figure 2.10, or they are located close to each other. In other words, it are the combination or fusion of the visual sensor with modern micro-controller technology. In an architectural view, this kind of camera is represented in two layers: (*i*) Visual sensors layer and (*ii*) Mist computing layer.

Visual sensors layer

This layer includes visual sensors, which can capture the events located in their FoV and send them to their corresponding mist node to analyse them.

Mist layer

Mist computing pushes processing even further to the network edge, decreasing latency and increasing subsystems' autonomy. In our context, every camera's self-awareness is critical, as the computation and actuation are dependent on the camera's perception of the situation. By adopting the mist paradigm, it is possible to give all the cameras of the IVSS cognitive intelligence to speed up the recognition and analysis tasks.

In our proposed system, micro-controllers represent the kernel of a smart camera. Thanks to the latter, the cameras become intelligent and able to process incoming data from the visual sensor and/or share it with other cameras or nodes in the next Layer. In our scenario, the first task assigned to each mist node is detecting objects from the incoming frames (i.e., identifying objects of interest in the scene). Then, the mist node extracts each detected object's feature vectors and sends them to the next layer to classify and identify them. The mist node also sends the raw data to the fog layer. After identifying the detected objects, the mist node starts the tracking processes until the tracked object leaves the camera's FoV.

Fog layer

Fog computing has many benefits for the video surveillance system that encourages us to integrate it into our system. Among these benefits, we briefly highlight the following:

- ✓ *Low latency:* The fog layer consists of several fog nodes near the surveillance cameras (to the edge of the surveillance network) to reduce data transmission latency and pre-process like filtering raw data before sending it to the next layer.
- ✓ *Save network bandwidth:* Fog computing can also store time-sensitive data and make an intelligent decision at the fog node, avoiding many data transmission transactions from/to the next layer and, consequently, saving bandwidth.
- ✓ *Large number of nodes:* The fog is suitable to build coherently a distributed video surveillance system based on the large-scale multi-camera that requires distributed computing and storage resources due to wide geo-distribution. Fog computing meets the requirements thanks to the possibility of connected thousands or more cameras in an extensive distributed video surveillance system, as in smart cities.
- ✓ *Real-time interactions:* The fog eliminates the problem of real-time interactions because it operates at the edge of the network. As a result, fog can significantly contribute to the hard-critical real-time processing of video surveillance systems.

- ✓ *Heterogeneity:* Fog nodes come in various forms and can be deployed in a wide variety of areas. Multiple manufacturers can provide various forms of smart cameras, and the camera can be deployed in a different environment of the same surveillance system. Therefore, the heterogeneous characteristics of fog will provide utility in video surveillance applications.
- ✓ *Interoperability:* The fog component is typically capable of communicating with each other to distribute resources across the network. Therefore, it will help support seamlessly support applications such as object identification, which requires the cooperation of various fog nodes. Consequently, the fog paradigm can help video surveillance applications, as next-generation video surveillance systems often interact intelligently with other external subsystems or systems.

Each fog node manages and controls a set of smart cameras located in their geographical area in our system. After the fog node receives the data (feature vectors and raw data) transmitted by the mist nodes, it stores the feature vectors to identify (labelling) the requested objects and be shared with other cameras or neighbouring fog nodes if necessary. Concerning the raw data, they are transmitted to the upper layer.

Cloud layer

The fourth layer is the cloud computing layer. The cloud or datacenter is adopted for the following reasons: (*i*) the large storage capacity to save the raw system data transferred via the lower layer, and (*ii*) the computing power to execute the robust algorithms like analysing objects' behaviour. In our system, the cloud's role is to observe and analyse human behaviour to identify any inappropriate or suspicious behaviour in the monitored area (see Figure 4.3) based on the data collected from the fog layer. In this phase, artificial intelligence, machine learning and deep learning are the main techniques used to detect this type of behaviour with precision and speed unattainable by the human eye alone [26, 162].

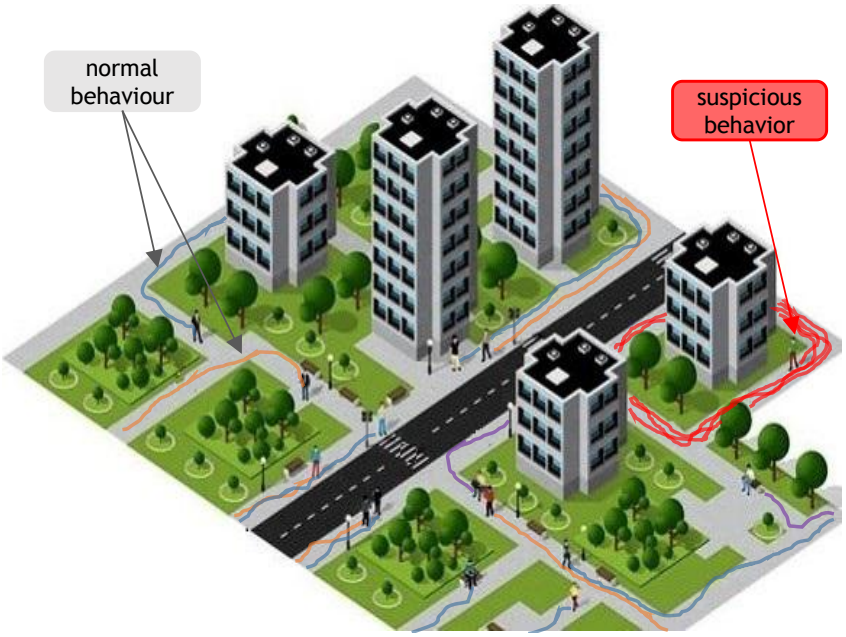


FIGURE 4.3: Example of normal and suspicious behaviour

Application layer

The last layer is the Application layer. It allows the management of the control applications of the system by the end-users. This layer is represented by the control office and managed by qualified human operators (police or private security officers). Their role is to monitor the system in the case of a failure or emergency (such as detecting suspicious objects) and provide services such as tracking the traces of a given object (see Figure 4.3).

4.3 The proposed strategy for multi-camera clustering

Over the past decade, researchers have paid particular attention to multi-camera networks to develop a more efficient, flexible, and cost-effective surveillance system adapted to emerging applications for smart city construction based on IoT [167, 239] or, more precisely, based on IoVT. Surveillance systems consist of a set of intelligent cameras distributed in the surveillance zone. These cameras' role focuses on extracting visual data captured in the environment and sharing it with other cameras or computer centres to analyse and process them for tracking or re-identifying objects.

In conventional systems, the cameras are autonomous; operate without any knowledge of these neighbours' distribution or functioning because of the absence of communication and coordination between them. This lack leads to redundant event detection, which means more redundant multimedia data is generated and, therefore, more computing power, more storage space, and more bandwidth to process, store and transmit it. All this leads to a rapid increase in energy consumption. To reduce redundant data and the energy consumed, the researchers have introduced a new surveillance system that includes intelligent collaborative multi-camera. This new system allows the coordination of the cameras' operation through communication to achieve common goals with efficient performance and the lowest possible cost.

According to Prabhu's definition in subsection 2.4.4, collaboration in the surveillance camera network means a shared volume of data between the network cameras. In large-scale multi-camera networks, the number of cameras is higher. Therefore, the amount of data shared increases exponentially with the number of cameras. This exponential increase can lead to network overload or congestion, leading to network saturation due to the large amount of data transmitted over the network, which harms the system's reliability. One of the main solutions to overcome these problems is clustering and camera planning. Clustering cameras have several objectives, namely [19, 50, 51]: **(i)** network scalability, **(ii)** reduction of power consumption to extend battery and network life, **(iii)** stabilization of network topology, **(iv)** reduction of overhead costs.

In multi-camera networks, especially in dense networks, the cameras' FoVs may overlap, which means that common sub-areas are resulting in network energy loss due to redundant overlap detection. We propose a new clustering method based on the FoV overlap surface criterion instead of the radio or distance criterion between cameras to address this problem. Our main idea is that if the area of FoV overlap between the cameras is relatively large, the cameras will act similarly in terms of coverage. Therefore, we will group them into a single group. Our approach's main objective is to create highly overlapping groups of cameras to restrict communication and coordination only at the group level and not for all system cameras. In this way, our approach can significantly simplify the coordination process and avoid the redundant detection of events between cameras after finding that overlapping cameras cause this problem (RQ2).

Consequently, it plays a vital role in conserving energy and enhancing the network's lifetime by enabling collaboration capabilities between the group's cameras and avoiding redundant detection and processing.

4.3.1 Related works

Several studies in the literature use clustering algorithms in wireless sensor networks (WSNs) to develop eco-energetic routing protocols to increase network monitoring life [155, 255]. Abbasi and Younis [4] survey current clustering mechanisms for scalar sensor networks and provide the primary keys for designing such algorithms. Most of the clustering algorithms group nodes based on sensor neighbourhood or the distance—in meters or hops—from a node to the cluster-head (CH). The number of clusters and cluster-size are parameters that usually impact cluster formation procedures. However, Obrackza et al. [312] state that directional FoV should be the critical parameter to form clusters in WMSNs and highlight examples in which video sensor spatial-based collaboration provides robust object detection cross-validating information.

In WMSNs, Alaei et al. [17–21] propose several camera clustering algorithms for WMSNs based on overlapping areas between camera FoVs to establish cooperation between clusters that have been formed to detect objects. The objective of this work is to save energy and increase the life of the network. Their proposed methods are based on cluster members' random choice without considering the networks' overall contextual state. The overlap of cameras with several neighbouring cameras does not take into account. Common cameras, i.e., cameras that belong to more than one cluster, will deplete their battery fast. This latter is because several tasks are performed for each group to which it belongs, compared to other cameras that only perform the tasks performed in their group. Also, in their proposal based on membership in several clusters, if the number of common cameras increases, then the problem that we are grouping cameras for will reappear. The same principle is used by [93], where the authors propose a hierarchical clustering algorithm based on overlapping FoVs.

In [215], the authors present a clustering algorithm based on overlapping regions to reduce redundant video streams and reduce energy waste. The grouping of cameras in this technique is based on the correlation rate between the image captured by the cameras using the colour histogram difference approach. It detects potential scene cuts with a threshold to separate the cut scene from similar scenes and decide whether these differences are significant enough to identify a cut scene or not. This method is reliable, but only in areas with low overlap. The colour histogram is not efficient for grouping overlapping cameras because it represents the weighted average of the RGB colour components but not the content. Besides, light brightness strongly influences the histogram, making it very difficult to use this technique to measure the correlation between two images captured by two cameras with different FoV orientations. Moreover, using the colour histogram makes it possible to obtain two correlated images while the corresponding cameras are located further away from each other.

Shreya Mishra et al. [286] propose a method of clustering cameras according to the camera's communication radius to improve directional sensor networks' coverage. The authors model the clusters by circles representing the communication range and select the first node as the cluster centre. Unfortunately, the cameras do not have a FoV in a circle, making this approach inapplicable outside the communication itself.

Danial et al. [113] propose an algorithm for selecting the minimum number of cameras activated to cover all targets. They take into account the multiple views of the targets to calculate

redundancy in the WVSN network. This approach requires computational power at the camera level to ensure the recognition mechanism that in most cases fails for multiple reasons such as the high number of targets, the complexity of the environment, and the performance limits of surveillance cameras.

KyDong Jung et al. [205] propose a clustering algorithm called FL-TEEN that uses a fuzzy inference system to improve cluster head selection's adaptability to enhance performance in terms of sensor node lifetime. This semi-automatic approach requires human intervention to generate rules for each environment. To solve the problem of region coverage, Selina Sharmin et al. [376] have developed a system of area coverage sensitive to the network's life that uses a clustering mechanism based on the degree of overlap and residual energy levels. This proposal favours network management in the event of a sensor failure but at the expense of monitoring quality.

4.3.2 Clustering multi-camera based on FoV overlaps: Global strategy

The general strategy of our approach is illustrated in Figure 4.4. The strategy is based on the system's load distribution at the fog computing level presented in the proposed architecture in section 4.2 to reduce the task's complexity and take advantage of all the resources available in the system. Each fog node is responsible only for clustering cameras located in their region or geographical area. Our strategy's scenario is started after the cameras' deployment in the surveillance area. Each camera sends its information (location, detection range, vertex angle, etc.) to the corresponding fog node. Based on this information, the fog node will initially start the camera clustering phase. As shown in Figure 4.4, this phase consists of three main steps, namely: **(1)** determination of the FoV intersection polygon for two cameras having an overlap between them, **(2)** calculation of the surface area of each polygon, **(3)** finally, application of one of the two proposed methods that we will describe in subsection 4.3.2.3 to group the cameras that have a significant overlap area. While the system runs, it may face unforeseen changes that need to be processed in real-time, such as a camera breaking down or adding a new camera to the system. In this case, the fog node moves to **(4)**, the update phase, to manage these exchanges. Our strategy is executed only once, just after the cameras are deployed.

4.3.2.1 The overlap area determination step

The FoV of a camera is defined by the area in which a camera can easily and accurately detect objects covered by the latter. According to [19], we can model the FoV by the surface of an isosceles triangle (ABC) as shown in Figure 4.5. The vertex A is considered as the position of the camera; the two other vertices are calculated by the Equations 1, 2, 3 and 4, as follows:

$$X_B = X_A + R_s \cdot \cos(\alpha) \tag{1}$$

$$Y_B = Y_A + R_s \cdot \sin(\alpha) \tag{2}$$

$$X_C = X_A + R_s \cdot \cos((\alpha + \theta) \bmod 2\pi) \tag{3}$$

$$Y_C = Y_A + R_s \cdot \sin((\alpha + \theta) \bmod 2\pi) \tag{4}$$

In our modelling, the overlapping FoV between two cameras are irregular polygons. To determine these polygons, we must first find all the polygons vertices represented by the two triangles' FoVs intersection points. Figure 4.6 shows some examples of the intersection of two FoVs.

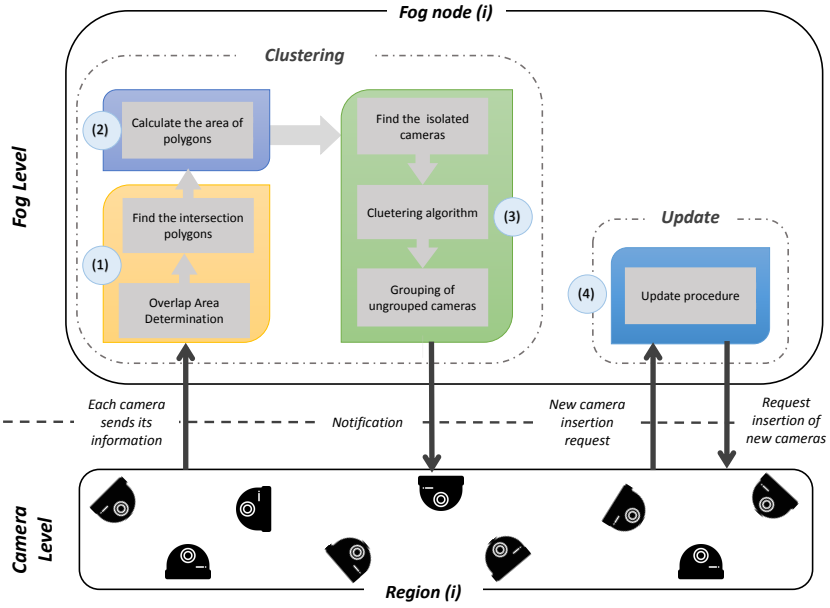


FIGURE 4.4: The steps of clustering strategy

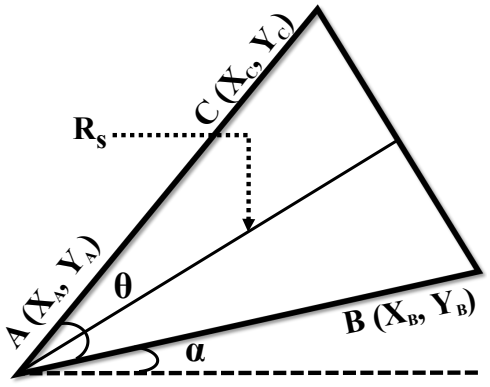


FIGURE 4.5: FoV of camera sensor (from [18])

As shown in Figure 4.6, two types of vertices are possible: 1) vertices generated by the triangle edges intersection and 2) vertices representing the triangle's vertex included in another triangle. First, it is necessary to look for the vertices of the triangles inside the other triangles. For this purpose, we used Equation 5 and conditional expression 6.

Let T be a triangle with the vertices $\mathcal{A}(x_1, y_1)$, $\mathcal{B}(x_2, y_2)$ and $\mathcal{C}(x_3, y_3)$. $\mathcal{M}(x, y)$ is a point in space. $m, k \in \mathcal{R}$

$$\begin{cases} x &= k \cdot (x_2 - x_1) + m \cdot (x_3 - x_1) + x_1 \\ y &= k \cdot (y_2 - y_1) + m \cdot (y_3 - y_1) + y_1 \end{cases} \tag{5}$$

$$\begin{cases} \mathcal{M} \in \mathcal{ABC}, & \text{if } m \geq 0 \wedge k \geq 0 \wedge m + k \leq 1 \\ \mathcal{M} \notin \mathcal{ABC}, & \text{Otherwise} \end{cases} \tag{6}$$

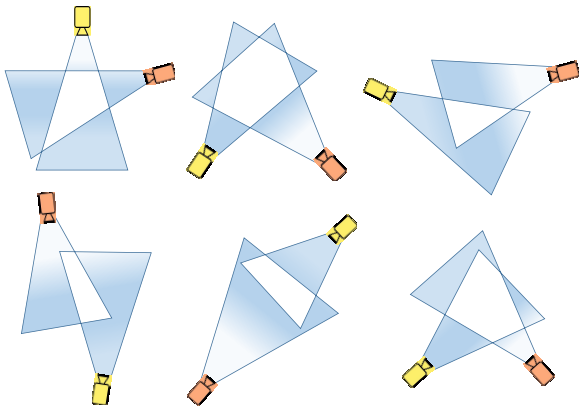


FIGURE 4.6: Different possibilities for FoV overlapping

If there is a vertex among the set of the vertices of the triangles that satisfy conditions 6, these vertices' coordinates will be added to the polygon points list.

The next step is to determine the intersection points between the sides of the triangles (two by two) by solving equation 7. The conditional expression 9 was used to check whether the point of intersection is accepted or not as the vertex of the polygon.

Let $(\mathcal{AB})$ and $(\mathcal{CD})$ be two lines with the following equations:

$$\begin{cases} (\mathcal{AB}) : y &= ax + b \\ \\ (\mathcal{CD}) : y &= a'x + b' \end{cases} \tag{7}$$

With, $a, b \in \mathcal{R}$, where $(a, b) \neq (0, 0)$ and $a', b' \in \mathcal{R}$ where $(a', b') \neq (0, 0)$.

$$\begin{array}{llll} ((x_A \leq x \leq x_B) & \wedge & (x_C \leq x \leq x_D)) & \vee \\ ((x_A \leq x \leq x_B) & \wedge & (x_C \geq x \geq x_D)) & \vee \\ ((x_A \geq x \geq x_B) & \wedge & (x_C \geq x \geq x_D)) & \vee \\ ((x_A \geq x \geq x_B) & \wedge & (x_C \leq x \leq x_D)) & \\ & & & \\ ((y_A \leq y \leq y_B) & \wedge & (y_C \leq y \leq y_D)) & \vee \\ ((y_A \leq y \leq y_B) & \wedge & (y_C \geq y \geq y_D)) & \vee \\ ((y_A \geq y \geq y_B) & \wedge & (y_C \geq y \geq y_D)) & \vee \\ ((y_A \geq y \geq y_B) & \wedge & (y_C \leq y \leq y_D)) & \vee \end{array} \tag{8}$$

$$\begin{cases} P(x, y) \text{ accepted} , & \text{if (8) is checked} \\ P(x, y) \text{ unacceptable} , & \text{otherwise} \end{cases} \tag{9}$$

The coordinates of each accepted intersection point have also been added to the list of polygon points. At the end of this step, all the polygon points are detected and saved in a list for further processing.

4.3.2.2 The polygon area calculation step

The next step is to calculate the area of the polygons using equation 10 :

Let $Gon(x_i, y_i)$, $i = 0, ..., n$ a polygon, where n is the number of polygon vertices such that $(x_0, y_0) = (x_n, y_n)$.

$$A = \frac{1}{2} \sum_{i=0}^{n} (x_i \cdot y_{i+1}) - (x_i \cdot y_{i-1}) \tag{10}$$

Each polygonal surface generated by the intersection of the fields of view of two cameras is recorded in the surface matrix $\mathcal{S}[n \times n]$, where n is the number of cameras. For example, the polygonal area $a_{i,j}$ generated by the intersection between cameras with identifier i and j, respectively, is added in the cell $\mathcal{S}[i, j] := a_{i,j}$. After calculating all surfaces, a symmetrical square matrix with a zero diagonal was obtained, as shown in Figure 4.7.

	0	1	2	3	4
0	0	...	...	...	...
1	...	0	$S_{(1,2)}$	...	...
2	...	$S_{(1,2)}$	0	...	...
3	...	...	...	0	...
4	...	...	...	...	0

FIGURE 4.7: The matrix of surfaces $\mathcal{S}[5 \times 5]$

4.3.2.3 Clustering step

After calculating all the intersection polygons' surfaces in the previous step, we move on to the cluster (definition 4.1) extraction step.

Definition 4.1 (Cluster)**.** A cluster represents a subset of cameras with overlapping FoV. In our context, the degree of overlap between the cameras' FoVs determines whether they can be in the same cluster (grouping criterion).

At first, the fog node finds the isolated cameras (see definition 4.2) and puts each camera in a single cluster. This first step is carried out to reduce the complexity of the grouping process because isolated cameras are never grouped with another group due to the lack of overlap with other cameras.

Definition 4.2 (Isolated camera)**.** A camera is considered isolated if and only if all overlapping surfaces with other cameras are null.

Let $\mathcal{C}$ a set of cameras where $\mathcal{C} \neq \varnothing$. Let $c_i \in \mathcal{C}$ a camera. $\delta(\cdot, \cdot)$ is a function to calculate the overlap between two cameras. c_i is isolated camera iff $\sum_{j=1}^{n} \delta(c_i, c_j) = 0$, where $n = |\mathcal{C}|$ and $i \neq j$.

Each fog node executes the clustering algorithm, which takes the surface matrix as an input and returns the list's camera groups. At this stage, we propose two methods for grouping the cameras, and they are explained in the following.

▷ Proposal 01: Clustering based on the ascending hierarchical classification (AHC) algorithm

❑ *Hierarchical classification*

Hierarchical classification is a set of unsupervised classification or clustering techniques. These techniques use an iterative process to regroup or redistribute original data. They are based on the selection of an aggregation criterion to determine how to agglomerate two clusters or divide a cluster, i.e., groups of objects such as:

- ❖ The objects are as similar as possible within a group (compactness criterion).
- ❖ Groups are as dissimilar as possible (separability criterion).
- ❖ Resemblance or dissimilarity is measured on all descriptive variables.

What interests the analyst is not the hierarchy, but a typology, i.e., a partition of the objects into clusters that are: (*i*) compact, (*ii*) well separated from each other, and (*iii*) easily interpretable.

❑ *Ascending hierarchical classification -AHC-*

In AHC (algorithm 4.1), the process starts by finding among the n clusters −with each cluster formed by a single object (singleton clusters)− the two most similar clusters for all the p specified variables. It will then merge these two clusters to form a new cluster. Thus, at this level $(n-1)$ clusters, one comprises two previously grouped clusters, the others containing a single object. The process continues by determining the two clusters that are the most similar and merging them until a single cluster is obtained with all the objects. The methods are distinguished by the distance between observations and the definition of the aggregation strategy.

Algorithme 4.1 The basic algorithm of the AHC (from [120])

1: Let $\mathcal{I} = \{c_1, c_2, \cdots, c_n\}$ a set of clusters ▷ At the beginning are singletons clusters
2: Calculate the distances $d(c_i, c_j)$ for any pair (c_i, c_j) of clusters of $\mathcal{I}$
3: Find c_i and $c_j \in \mathcal{I}$ such that for all $\{c'_i, c'_j\} \in \mathcal{I} : d(c_i, c_j) \leqslant d(c'_i, c'_j)$
4: **for** the couple (c_i, c_j) **do**
5: Form a new cluster $c_{new} = c_i \cup c_j$
6: $\mathcal{I} := \mathcal{I} - \{c_i, c_j\}$ ▷ Remove c_i, c_j from $\mathcal{I}$
7: $C = \{c_{new}\}$
8: **end for**
9: **if** $|\mathcal{I}|$ **is** $\varnothing$ **then** ▷ All objects are grouped
10: **break**
11: **end if**
12: **for** $c_i \in \mathcal{I}$ **do**
13: Calculate the distance $d(c_i, c_{new})$ using the chosen aggregation formula
14: **end for**
15: $\mathcal{I} := \mathcal{I} \cup C$
16: **go to** line 14

Many aggregation criteria have been proposed. The simplest and the most well-known are the following:

- *Single-linkag*: The distance between two classes C_1 and C_2 is defined by the shortest distance between an individual from C_1 and an individual from C_2.

$$d(C_1, C_2) = \min_{x_1 \in C_1, x_2 \in C_2} (d(x_1, x_2)) \tag{11}$$

- *Complete-linkage*: The distance between two classes C_1 and C_2 is defined as the greatest distance between an individual from C_1 and an individual from C_2.

$$d(C_1, C_2) = \max_{x_1 \in C_1, x_2 \in C_2} (d(x_1, x_2)) \tag{12}$$

- *Average linkage*: The average linkage is measured as the average of the distances between the individuals of C_1 and C_2.

$$d(C_1, C_2) = \frac{1}{n_1 \cdot n_2} \sum_{x_1 \in C_1}^{n_1} \sum_{x_2 \in C_2}^{n_2} (d(x_1, x_2)) \tag{13}$$

with $n_1 = |C_1|$ and $n_2 = |C_2|$ the cardinal of the two classes.

In our proposal, we focused on the AHC for the successive merge of cameras. Complete-linkage was chosen as grouping criteria due to its ability to generate small, homogeneous groups with large inter-group variability [75]. At each iteration, the two cameras with the largest overlapping area measurement are merged into a cluster. Initially, all cameras are considered as single-camera clusters. The first step involves grouping the two closest cameras, depending on the overlap area. Following an iterative process, our method continued to merge the most overlapping clusters while respecting the cut-off threshold β (see Figure 4.8) that was defined previously. The grouping process terminates when all cameras are grouped or overlapped areas are less than the specified threshold ($< \beta$).

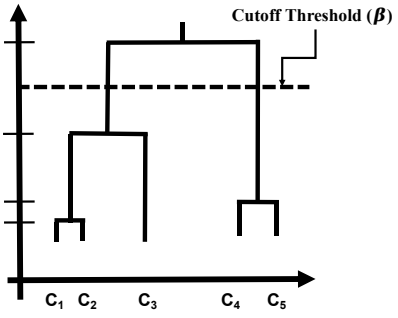


FIGURE 4.8: Example with cut-off threshold (β)

This strategy was applied on the surface matrix obtained at the end of the previous step (subsection 4.3.2.2), as presented by algorithm 4.2. When the execution of the grouping algorithm is completed, the fog node sends a notification to all cameras of their corresponding group ID and the cluster members.

The camera selection with better visibility is done in the detection and tracking step (section 4.4). Therefore, at this moment, the angle and range of the camera are used only to define the overlap limits. Indeed, these two criteria directly affect the quality of the results, but not on the

algorithm's behaviour. In other words, increasing the angle and range of cameras will reduce the number of groups and vice versa.

Algorithme 4.2 AHC-based cluster formation algorithm

1: **Input:**
2: $\mathcal{S}[n \times n]$: Matrix of overlapping areas (Figure 4.7)
3: β: Clipping threshold
4: $\mathcal{C}$: List of cameras
5: $\mathcal{SC}[\dots]$: State of the cameras. "$true$" means the camera is grouped and "$false$" otherwise.
6: **Output:**
7: $\mathcal{CL}$: list of clusters that contains the camera identifiers.
8: **function** $\mathcal{OV}_{max}(\mathcal{S})$ ▷ Function returns the two cameras that have the max overlap
9: $\forall c'_i, c'_j \in \mathcal{I}, \exists c_i, c_j \in \mathcal{I} : \mathcal{S}[i,j] \geq \mathcal{S}[i',j'] \wedge (\mathcal{SC}[i] = true \vee \mathcal{SC}[j] = true)$
10: **return** $\{c_i, c_j, \mathcal{S}[i,j]\}$
11: **end function**
12: **function** $\mathcal{IS}_{cam}(\mathcal{S}, \mathcal{SC})$ ▷ Function works to find isolated cameras
13: $\mathcal{I} := \varnothing$ ▷ $\mathcal{I}$: List of isolated cameras
14: $\forall c_i \in \mathcal{I}, \nexists c_j \in \mathcal{I} : \mathcal{C}[c_i, c_j] > (\beta \cdot S_{FoV})$ ▷ S_{FoV}: FoV surface
15: $\implies \mathcal{I} = \mathcal{I} \cup \{c_i\}$ *and* $\mathcal{SC}[i] := True$
16: **end function**
17: **Begin**
18: $\forall cs_i \mid cs_i \in \mathcal{SC} : cs_i := false$ ▷ All cameras are ungrouped
19: $\mathcal{CL} := \mathcal{CL} \cup \mathcal{IS}_{cam}(\mathcal{S}, \mathcal{SC})$ ▷ see the function in line (12)
20: $(c_1, c_2, \mathcal{MF}) := \mathcal{OV}_{max}(\mathcal{S})$ ▷ $\mathcal{MF}$: Maximum overlap value (see line (8))
21: **if** $\mathcal{SC}[c_1]$ and $\mathcal{SC}[c_2]$ are $false$ **then**
22: $C_{new} := \{c_1, c_2\}$ ▷ New cluster
23: $\mathcal{CL} := \mathcal{CL} - \{c_1, c_2\}$ *and* $\mathcal{CL} := \mathcal{CL} \cup \{C_{new}\}$
24: $\mathcal{SC}[c_1] := true$ *and* $\mathcal{SC}[c_2] := true$
25: **else if** $\mathcal{SC}[c_1] == false$ **then** ▷ Already grouped
26: $\mathcal{CL}[k] \cup c_2 \mid c_1 \in \mathcal{CL}[k]$ *and* $\mathcal{SC}[c_2] := true$
27: **else**
28: $\mathcal{CL}[k] \cup c_1 \mid c_2 \in \mathcal{CL}[k]$ *and* $\mathcal{SC}[c_1] := true$
29: **end if**
30: **if** $\forall cs_i == false \mid cs_i \in \mathcal{SC}$ **then**
31: **return** $(\mathcal{CL})$
32: **end if**
33: **go to line** 20
34: **End**.

❖ *Complexity of our clustering algorithm*

As shown in algorithm 4.2, the input is the overlapping surface matrix $\mathcal{S}[n \times n]$, which is a symmetrical square matrix (see Figure 4.7), where n is the number of cameras. The complexity of the algorithm is evaluated according to the number of overlapping cameras (i):

- ✓ In the case where there is no overlapping between n cameras, the number of i is equal to 0 $(i = 0)$, and the complexity is estimated by: $O(n) = \frac{n^2}{2} - n$.
- ✓ In the case where there is i overlapping $(i > 0)$ between n cameras, the complexity is given by: $O(n) = \varphi(i) \cdot (\frac{n^2}{2} - n)$, where:

$$\begin{cases} \varphi(i) = \varphi(i-1) + i \\ \varphi(0) = 1 \end{cases}$$

Since $\varphi(i)$ will always provide a constant value, the complexity can be simplified and represented by : $O(n) = k \cdot (\frac{n^2}{2} - n)$, where, $k \in \mathcal{N}$.

According to the final formula, the complexity always remains in a *quadratic* order $O(n^2)$ in the worst case.

▷ Proposal 02: using the Bron-Kerbosch algorithm

As we saw in the first proposal, our ACH-based clustering method focused on clustering cameras based on the area of overlap. It focused all the time on cameras that have maximum overlap. However, their knowledge about the network and all cameras' status remains partial because it does not consider other neighbours' overlaps.

In our second proposal, we try to use a technique that allows us to consider the global state of the camera network or, more specifically, the state of camera overlap in the network during clustering. This proposal tries to group as many overlapping cameras as possible using the Bron-Kerbosch algorithm to find maximum cliques representing the camera clusters.

❏ Bron-Kerbosch algorithm:

Maximal clique (definition 4.3) enumeration is a graph clustering method for finding all vertices with the most influence on a graph. Finding the maximum clique is one of the problems encountered in the analysis of data graphs. This is a NP-difficult problem for which suitable solutions must be designed in the case of large graphs.

The Bron-Kerbosch algorithm presented in [74] is one of the fastest [224] and most efficient [28] algorithms for finding maximum cliques in undirected static graphs. The Bron-Kerbosch algorithm is a recursive algorithm based on the backtracking technique to find the maximum cliques. In graph theory, a complete sub-graph is called a clique, and a maximum clique is a sub-graph in which no vertices could be added without losing the clique's property [368]. In this part, we explain the basic idea of the Bron-Kerbosch algorithm. Then, we present the use of this algorithm in our problem (camera clustering).

Definition 4.3 (Maximum clique)**.** Maximum clique is a clique (def.4.4) that cannot be extended by including other adjacent vertices (see Figure 4.9).

Definition 4.4 (Clique)**.** The clique is a complete subgraph, i.e., a subset of vertices all in a two-to-one relationship.

The Bron-Kerbosch algorithm (algorithm 4.3) receives three disjoint vertex-sets as input parameters: P, R, and X. The set R is a clique, and $P \cup X$ is the set of all vertices adjacent to every vertex in R. Each vertex in $P \cup X$ is a witness that the clique R is not maximal yet. The set P represents the vertices that have not been considered yet, whereas the set X includes all vertices that have already been taken into account in earlier steps. This set is used to avoid enumerating maximal cliques more than once. In each cell, the algorithm checks whether the given clique R is maximal or not. If $P \cup X = \varnothing$, then there are no vertices that can be added to the clique. Therefore, the clique is maximal and can

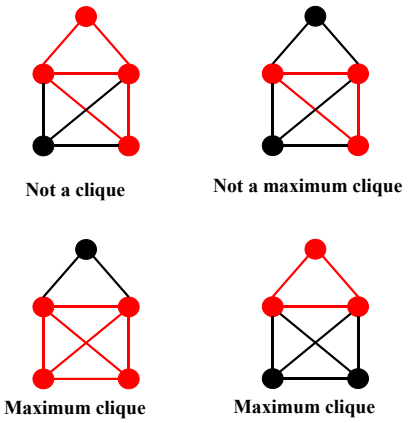


FIGURE 4.9: Example of clique and maximum clique

Algorithme 4.3 : Bron-Kerbosch algorithm

Input :
$P = \{v_1, v_2, \cdots, v_n\}$ ▷ Set of all vertices in graph G
$R = \{\varnothing\}$ ▷ Is a possibly a clique
$X = \{\varnothing\}$ ▷ Contains nodes already in some clique or processed
function BRONKERBOSCH(P, R, X)
  if $P \cup X = \{\varnothing\}$ **then**
    Report R as a Maximal Clique ▷ Add R to the solution
  end if

  for $\forall v \mid v \in P$ **do**
    BronKerbosch *(*$P \cap \{v\}$, $R \cup \{v\}$, $X \cap \{v\}$*)*
    $P = P \setminus \{v\}$
    $X = X \cup \{v\}$
  end for
end function

be added to the solution. Otherwise, the clique is not maximal because a vertex exists that is adjacent to all vertices in R and consequently would form a clique with R. For each $v \in P$ the algorithm makes a recursive call for the clique $R \cup \{v\}$ and restricts P and X to the neighborhood of v. After the recursive call, the vertex v is removed from P and added to X. This guarantees that the same maximal cliques are not detected multiple times. For a graph $G = (V, E)$ the algorithm will be initially called with $P = V$ and $R = X = \varnothing$ [186].

❑ Complexity

Bron and Kerbosch in [74] showed experimentally that the computing time per clique is almost independent of the graph size for random graphs and that the total computing time is proportional to $O((3.14)^{\frac{n}{3}}) \approx O((3)^{\frac{n}{3}})$ for *Moon–Moser* graphs[1] of n vertices [418].

In our context, a network of surveillance cameras can be considered as a non-oriented graph $G(E, V)$, where the cameras are simulated by vertices (V), and the arcs between the vertices

[1] *Moon–Moser* graphs: https://users.monash.edu.au/~davidwo/MoonMoser65.pdf

(E) represent the existence of overlaps between the cameras. A maximum clique or a complete sub-graphic corresponds to a cluster of cameras determined by the Bron-Kerbosch algorithm. Figure 4.10 present an example of our modelling. The three red nodes represent a maximum clique, and in the real system, the three cameras modelled by them represent a camera cluster.

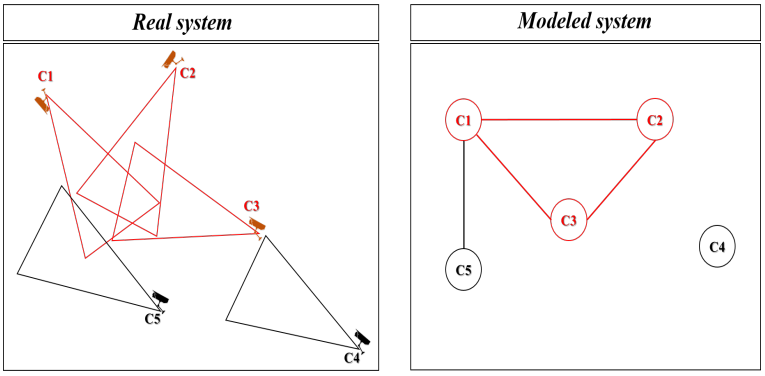


FIGURE 4.10: Modelling example

4.3.2.4 Update network

Our proposed strategy is executed by each fog node in the surveillance system once upon deployment, and thus all cameras will become grouped. If a camera joins the network hereinafter, it must send its information to the fog node corresponding to their region for announcing itself as a new camera. The fog node computes the FoV of the new node and finds the first cluster that can accept it as a new member. To do this, the fog node calculates the overlapping areas between the FoV of the new camera and the members of each cluster and checks whether it meets the cluster membership criteria. Then the fog node sends a message to the corresponding group members to re-organises the cluster with the new member. On the other hand, when a node dies, in this case, the CH camera will inform the rest of the cluster members of the new cluster set and reconfigure any cluster-related parameters.

4.3.3 Simulation and results

The results were obtained after executing the proposed clustering strategies on a workstation with an Intel® Core™ i5 with 4200U CPU, 1,6 GHz processor, and 4 GB RAM capacity. We developed their own simulator in JAVA to test the proposed approaches. The system operated without any constraints on the location and properties of the cameras. For this purpose, all cameras were randomly placed in the surveillance zone. The cameras were configured with an angle of FoV $\theta = 60°$ and $R_s = 25m$ in a sensing area of 300m * 300m. For our experiments, we use three types of cameras with low, medium, and high resolution, namely:

- Cyclops camera, which has a low resolution CIF (352x288) [341].
- MeshEye camera, which has a medium resolution SD (640x480) [182].
- SleepCAM camera, which has a high resolution 1080p HD (1920x1080) [278, 279].

Figure 4.11 shows the developed application. This simulator allows the user to create (Figure 4.11a) cameras with the previously mentioned features or to delete them (Figure 4.11d), and to modify cameras location (Figures 4.11b and 4.11c) according to the user's needs by the translation (Equation 14) or the rotation (equation 15) of the cameras.

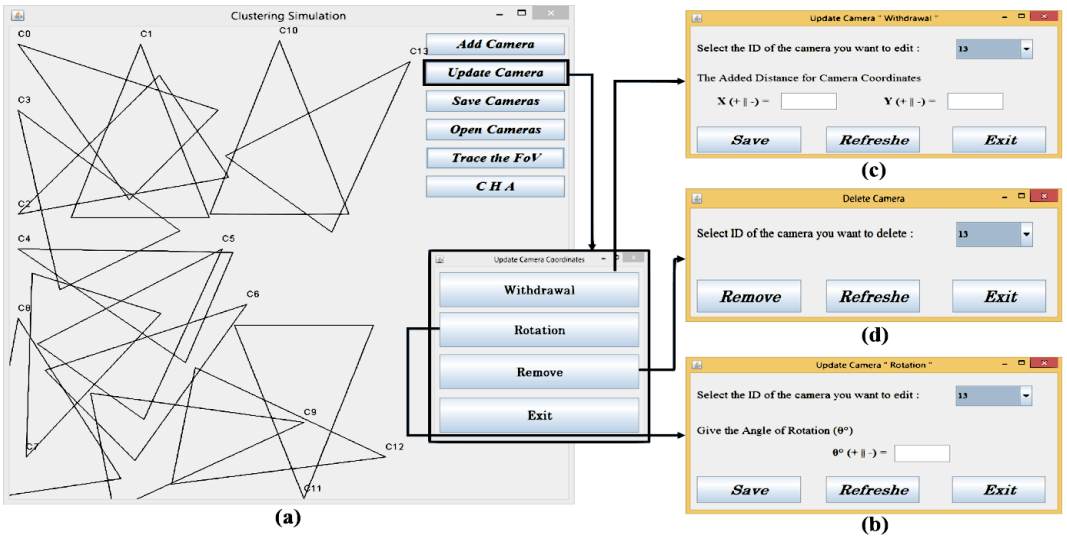


FIGURE 4.11: Simulator presentation

In the plan, the translation of the vector $\overrightarrow{u}(a,b)$, transforms the point $M(x,y)$ into $M'(x',y')$ as follows:

$$M'(x',y') = \begin{cases} x' &= x+a \\ y' &= y+b \end{cases} \tag{14}$$

Let $O(x_o,y_o)$ be the origin and θ, the rotation angle. The rotation transforms the point $M(x,y)$ into $M'(x',y')$ as follows:

$$M'(x',y') = \begin{cases} x' &= x\cdot\cos(\theta) - y\cdot\sin(\theta) + x_o \\ y' &= x\cdot\sin(\theta) + y\cdot\cos(\theta) + y_o \end{cases} \tag{15}$$

Figure 4.12 shows an example illustrating the different steps that the proposed system performs to group the cameras. Figures 4.12a and 4.12c present the first two steps: determination of the overlap area (intersection polygon) and calculation of the area of the polygons. Figure 4.12b shows the last step of grouping the clustered cameras, represented with the same colour, using the AHC-based clustering algorithm.

The system has been tested for six scenarios containing 50, 100, 150, 200, 250, and 300 randomly positioned cameras. Knowing that the results' quality is closely linked to the cameras' location, the system was tested 50 times for all cases. Each execution represents a random spatial dispersion of the cameras. As a result, the system was run 300 times.

As we mentioned before, the overlap threshold (β) determines the minimum overlap area of all cameras in the same cluster. However, β has a direct impact on the cluster camera selection

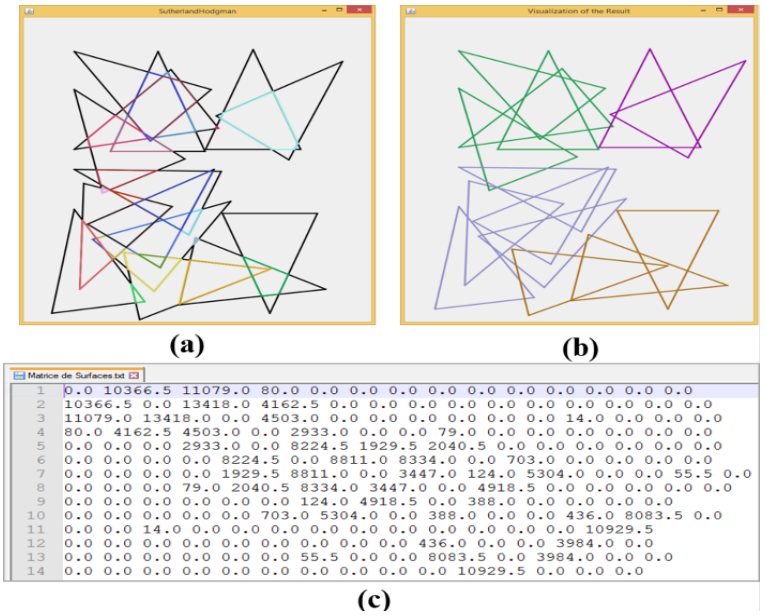
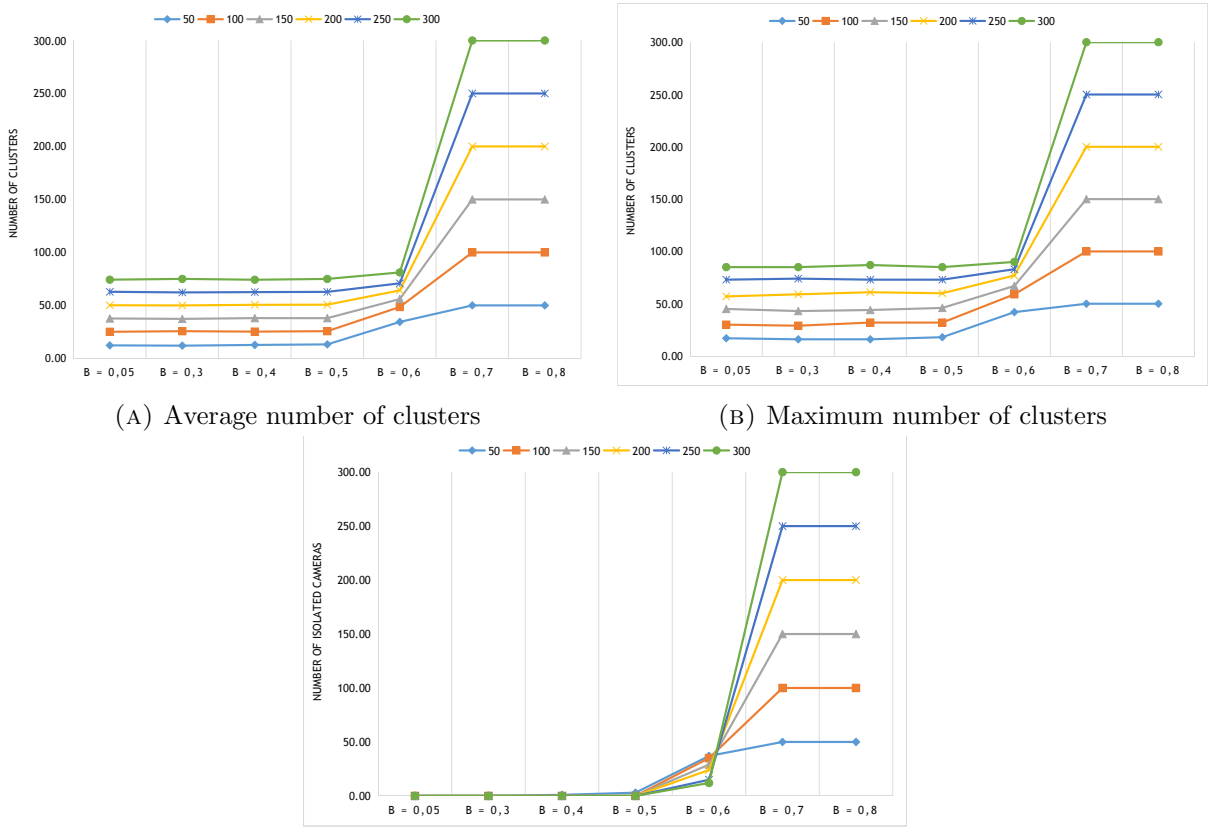


FIGURE 4.12: Simulation example (a) Intersection polygons (b) Clustering result (c) Surfaces matrix

process. To determine its adequate value, we will study this threshold's effect on our ACH-based clustering strategy. Consequently, Figures 4.13a and 4.13b show respectively the average and maximum number of clusters according to the value β.



(A) Average number of clusters

(B) Maximum number of clusters

(C) Number of isolated cameras

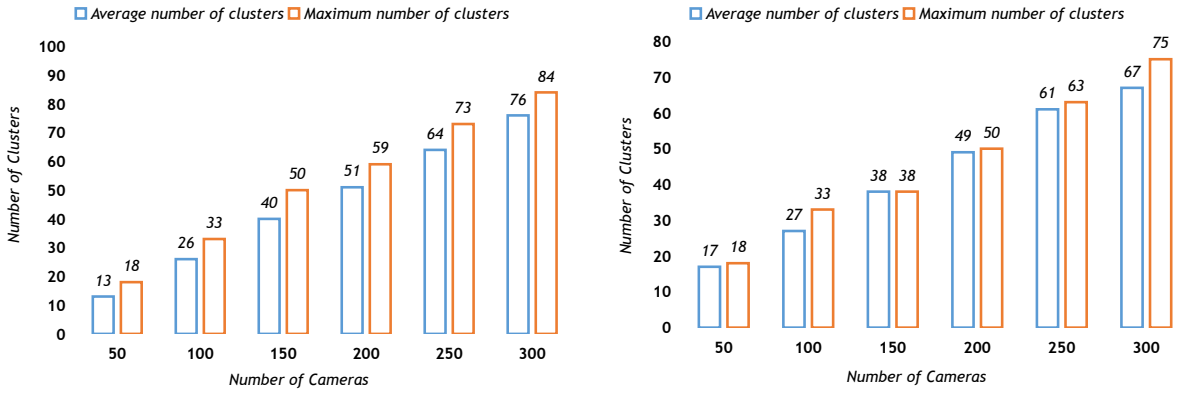
FIGURE 4.13: Evaluate the AHC-based method depending on the β value

Figures 4.13a and 4.13b demonstrate that β has an almost negligible impact for values in the range $\beta \in]0, 0.5]$ where the number of cluster results is stable. The average number of clusters is stabilised in 13, 26, 40, 51, 64, 76, respectively, for a network of 50, 150, 200, 250, 300 cameras. Concerning the values of the interval $\beta \in]0.5, 1]$, the number of clusters increases considerably until the number of clusters reached the number of cameras in the network. According to these results, we can conclude that the increase in β restricts camera membership due to the higher required overlap areas between the cameras' FoVs. Thus, higher overlap thresholds result in a higher number of clusters and a smaller cluster size. In other terms, we interpret the result that when the value of β increases by more than 0.5, the number of isolated cameras increases. To confirm this conclusion, we evaluated the number of isolated cameras in the same experiments, and the results obtained are shown in Figure 4.13c. A simple comparative observation of Figures 4.13a and 4.13b compared to Figure 4.13c, we can understand why the number of clusters reaches the total number of network cameras. It is confirmed that as the value of β increases, the number of isolated cameras increases. Therefore, the number of clusters containing a tiny number of cameras also increases. It remains that the value of β when is equal to 0. This case is the opposite that the value reaches 1. $\beta = 0$ means that there is no need to have overlapping areas to be a member of a group, and this has the consequence that all cameras are grouped into one group. In other words, β is totally negligible (see line 14 of algorithm 4.2).

The average and the maximum number of AHC-based clusters with $\beta = 0.5$ and Clique-based clusters are shown in the Figures 4.14a and 4.14b, respectively, while Figure 4.15 represents their grouping rat. The average of 300 cameras for the AHC-based clustering reaches 76 groups and up to 84 as maximum groups, with a clustering rate of 0.74%. In comparison, there is an

average of 67 groups and a maximum of 75 groups with a grouping rate of 0.77% for Clique-based clustering. For 200 camera cameras, cluster average and maximum numbers have been decreased to 51 and 59 clusters, and to 49 and 50 clusters for AHC and Clique-based clustering, respectively, with a rate almost equal to 0.75%.

A rating of 0,73% for the AHC-based clustering and 0,75% for Clique-based clustering was recorded for 150 cameras. For 50 cameras, the clustering rate was 0.75% and 0.65% for AHC- and Clique-based clustering, respectively. Therefore, it can be concluded that the proposed clustering methods achieved stability of approximately 0.75% and 0.74% for AHC- and Clique-based clustering methods.



(A) AHC-based method (B) Clique-based method

FIGURE 4.14: Average and maximum number of clusters

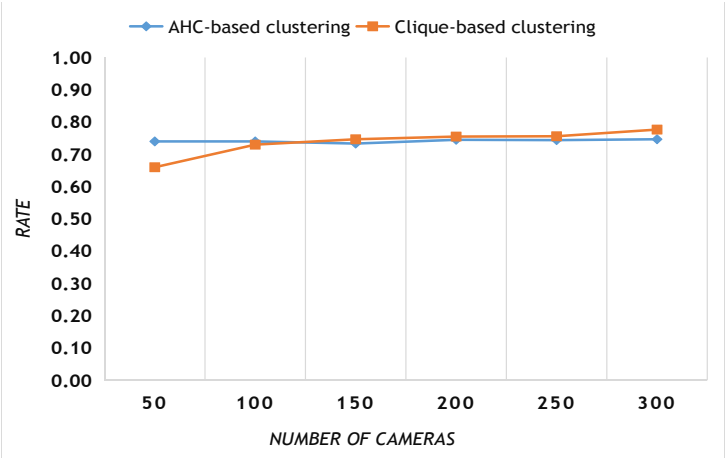


FIGURE 4.15: Clustering rate of AHC-based and Clique based method

Figure 4.16 illustrates that the number of cameras in the groups increases logarithmically with the number of cameras in the network for both methods. According to the observed results, an increase in camera density not only leads to an increase in the number of clusters (as shown in Figures 4.14a and 4.14b), but also to a wider overlap between the FoV of the cameras, which increases the size of the clusters. In the 300 camera test, the group size reached 6.22 cameras for the AHC-based clustering method and 7.75 for the Clique-based clustering method. In the 50 camera network, the group size reached 4.2 and 4.77 for the ACH- and Clique-based clustering methods, respectively.

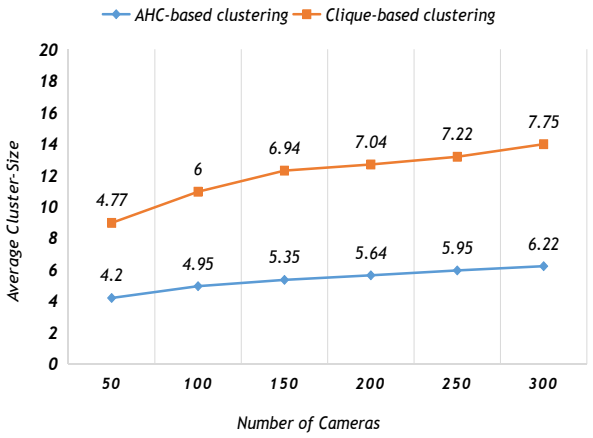


FIGURE 4.16: Average of cluster-size

To evaluate the proposed clustering method in terms of energy, we use the same camera coordination mechanism proposed in [18]. Alaei et al. propose an algorithm for the internal collaboration of cameras of the same groups generated by clustering formation. The algorithm works at two levels : (i) the lower level, which includes scalar sensors like motion detection sensors, and (ii) the upper level, which includes visual sensors (cameras). The algorithm starts at the lower level when scalar sensors detect movements in their detection range. These sensors warn the cameras located in the communication range that they have movements in their area. In the beginning, all cameras (upper level) are in a sleep state. When the cameras receive a message from the scalar sensors, each group's cameras, in turn, trigger the periodic motion detection according to algorithm 4.4.

Algorithme 4.4 : Cluster-based Cooperative Scheduling algorithm (from [18])

1: **for** $\forall\ C_j \in \mathcal{CL}$ **do** ▷ all clusters in parallel
2: i := 0 ▷ start with the first camera of each cluster
3: Wake up camera number i
4: Capture an image and then call object *detection*()
5: **if** *detection*() == *true* **then**
6: Send the image to sink ▷ In our context, the sink represents a fog node.
7: **end if**
8: $i := i + (1 \mod C_{size})$ ▷ select next node of the cluster
9: go to line 3
10: **end for**

As reported by the algorithm 4.4, in each cluster, the cameras are awakened sequentially intermittently by the first camera during a time interval $T = T_{interval}$ which represents the time between the awakening of two consecutive cameras (see Figure 4.17). This interval correlates with the size of the cluster, therefore, the degree of overlap of clusters, and $T_{interval} = \frac{T_p}{C_{size}}$, where T_p is the total operating time of the system. In this way, each camera of a given cluster periodically participates in the execution of the tasks shown in Figure 4.18 and finally sleeps with a period determined according to the size of the cluster to which it belongs (next interval). With this strategy, the energy consumed during the period $T_{interval}$ is reduced compared to the energy consumed by n cameras without this coordination.

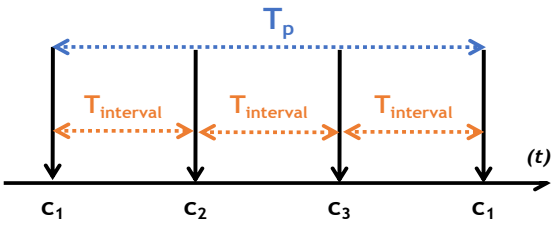


FIGURE 4.17: Scheduling for a cluster consisting of three cameras (c_1, c_2, c_3)

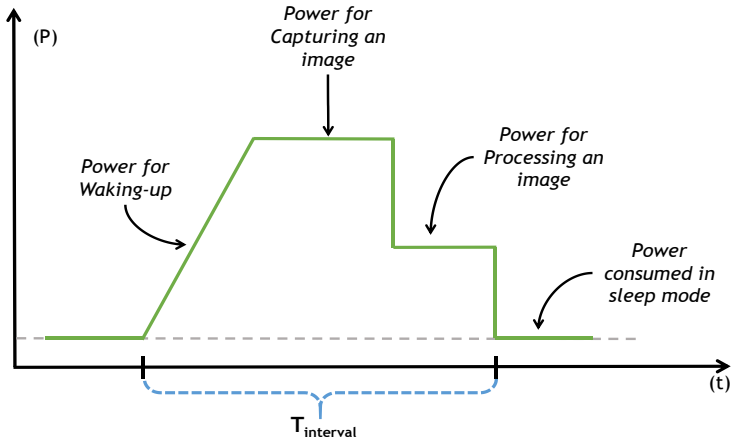


FIGURE 4.18: The main tasks of the camera in surveillance mode

From the previous Figure, we can define the energy consumed by a camera in $T_{interval}$ in active mode as the total energy required to wake up, capture an image, and process it. This is illustrated in Equation (16), where E_{up} and E_{cap}, and E_{proc} are, respectively, the consumed energies to activate a camera, capture an image and perform the desired task.

$$E_{camera}^{active} = E_{up} + E_{cap} + E_{proc} \tag{16}$$

For each $T_{interval}$ only one camera wakes up in each group. This means that in $T_{interval}$, there is only one active camera, and the other $(n-1)$ cameras are in sleep mode. Therefore, the total energy consumed by a single group in $T_{interval}$ is the energy consumed by the active camera and the energy consumed by the other cameras in sleep mode. The energy consumed by the cameras belonging to the cluster of the size equal to C_{size} during each interval is estimated by Equation (17), where E_{sleep} is the energy of a camera in sleep mode.

$$E_{cluster} = E_{camera}^{active} + (C_{size} - 1) \times E_{sleep} \times T_{interval} \tag{17}$$

The number of activated cameras in the network is equal to the number of clusters because, in each cluster, only one camera is activated at a time. The energy conservation ratio (ECR) for each cluster was defined as the ratio of the total amount of energy consumed by the nodes belonging to the cluster during each $T_{interval}$ in two ungrouped and grouped cases (see Equation (18)).

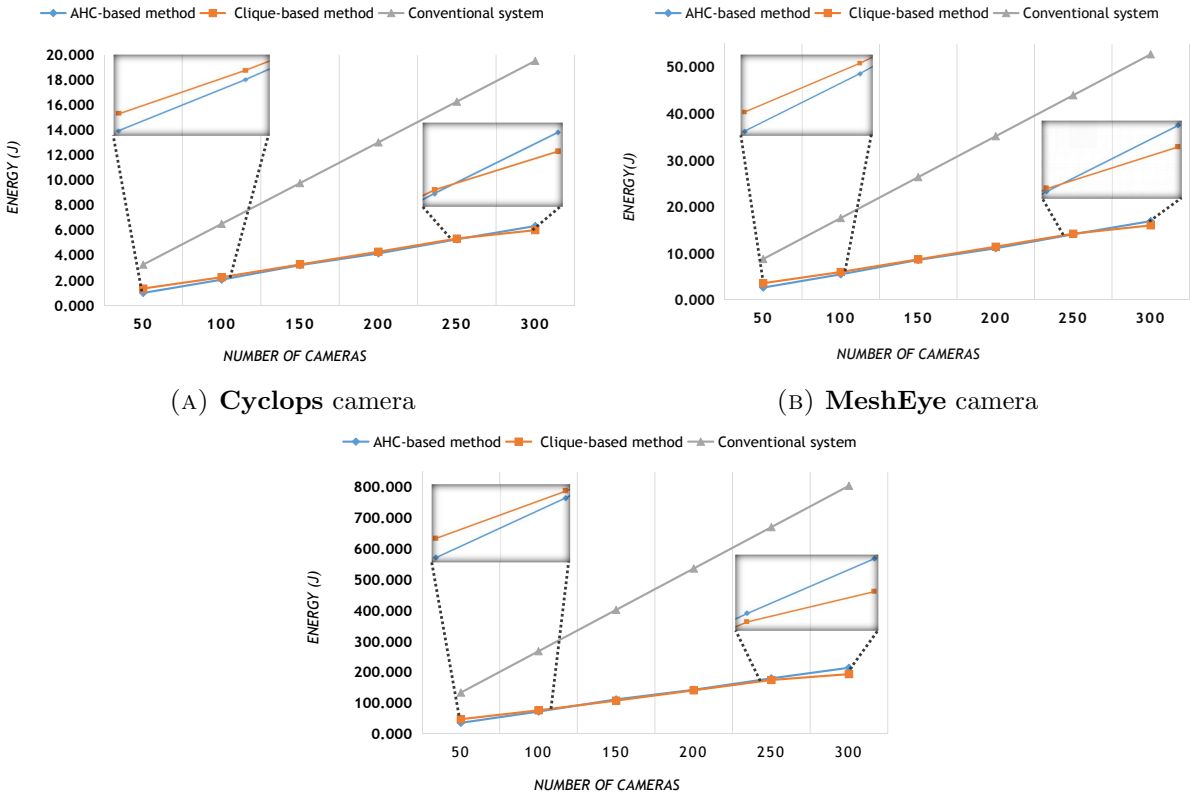
$$ERC = \frac{C_{size} \times E_{camera}^{active}}{E_{cluster}} \tag{18}$$

Clusters with a larger number of cameras save more energy and have a higher RCE since an overlapping region is shared among more coordinated members. For a network composed of C_{size} cameras, the average ECR ($AvgECR$) can be defined depending on the average cluster size ($AvgC_{size}$) as shown in that Equation (19):

$$AvgECR = \frac{AvgC_{size} \times E_{camera}^{active}}{AvgE_{cluster}} \tag{19}$$

Where,

$$AvgE_{cluster} = E_{camera}^{active} + (AvgC_{size} - 1) \times E_{sleep} \times T_{interval}$$



(A) **Cyclops** camera

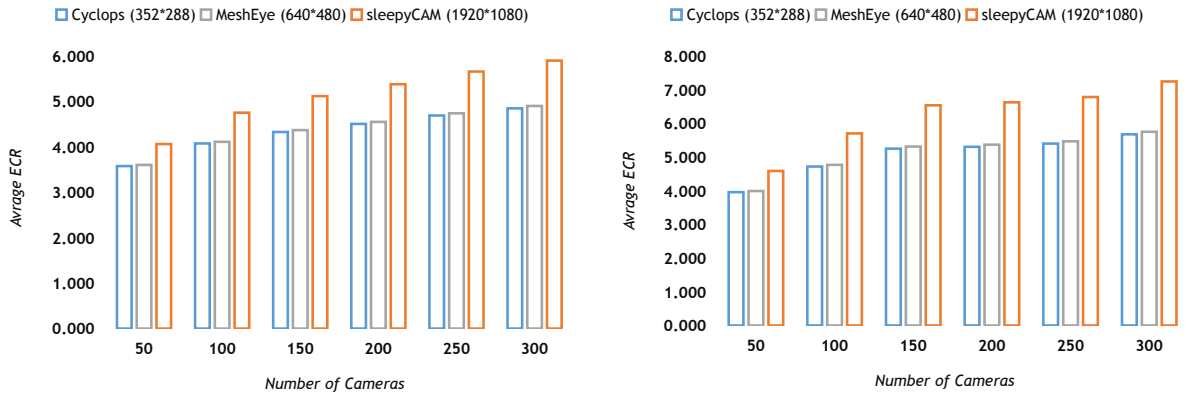
(B) **MeshEye** camera

(C) **sleepCAM** cameras

FIGURE 4.19: Estimated average energy consumption

Figure 4.19 show respectively the variation of the average energy consumed according to the number and type of cameras with and without clustering: *Cyclops* "CIF (352x288) low resolution camera" (Figure 4.19a), *MeshEye* Camera "SD (640x480) medium resolution" (Figure 4.19b), and *SleepCAM* Camera, "1080p HD (1920x1080) high resolution" (Figure 4.19c). Test results are obtained $T_{interval} = 5s$.

The results show that systems based on the clustering strategy consume less energy than conventional systems. The difference rate between them increases considerably with the increase in



(A) AHC-based method (B) Clique-based method

FIGURE 4.20: Average energy conservation ratio (AvgECR)

the number of cameras in the network and the increase in the cameras' resolution. In a network of 50 Cyclops cameras (low resolution), the rate is reached 59% for the Clique-based approach. This value is increased with a network of 300 Cyclops cameras until 70%, and up to 76% with a network of 300 SleepyCAM cameras (high resolution).

With a simple comparison of the energy consumed between the two grouping strategies, we can observe that the two methods have almost the same energy consumed. But with in-depth analysis, we find that the AHC-based strategy consumes less energy than the Clique-based strategy for a network of n cameras, where $n \in]0, 150]$ and vice versa in the opposite case (i.e., $n \in]150, 300]$). The latter is correlated with the clustering rate explained above (see Figure 4.15). This means that if the number of clusters is decreased (or the number of cameras is increased), the system will consume less energy.

Figure 4.20 presents the average energy conservation rate (AvgECR) between the two systems using the three types of cameras mentioned above. The AHC-based method results are shown in Figure 4.20a and the Clique-based method results are shown in Figure 4.20b. The results show the AvgECR increases slightly when the number of cameras with low or medium resolution increases, but it increases substantially when the number of high-resolution cameras increases. Between Figures 4.20a and 4.20b, we observe that the Clique-based strategy has a higher AvgERC compared to the AHC-based strategy, and this is due to lower energy consumption for the Clique-based strategy as we discussed above.

4.3.4 Summary

In this section, we have proposed two methods of clustering visual sensors/cameras based on the degree of overlap between cameras as a clustering criterion rather than the distance criterion used for scalar sensors. For system load distribution, the proposed strategy is executed at the fog computing level. Each fog node is responsible for grouping all cameras located in its geographical area (see Figure 4.4). The strategy execution is triggered after the cameras have been installed in the surveillance area, where each camera sends its information (location, detection range (R_s), and vertex angle (θ)) to the corresponding fog node. Once all camera information has been received, the fog node starts computing the overlap between the cameras. It then executes the appropriate clustering algorithm based on the degree of overlap between the cameras. At the end of this strategy and after the camera groups have been created, each fog node sends to

each camera its group ID and the neighbouring cameras (i.e., cameras that belong to the same group).

The first proposed technique is based on the ascending hierarchical classification (AHC) algorithm. This algorithm always starts with the grouping of strongly overlapping cameras. However, this technique has partial knowledge of the network and its state, i.e., it only knows the maximum overlap. The other overlaps are not taken into account (i.e., are completely neglected).

For this purpose, we have proposed a second clustering technique that allows us to regroup not only the two cameras that overlap the most, but also all cameras that overlap with as many cameras as possible. This principle is adequate to find the groups we want according to the answer we are given. To find this type of group, we model the network of cameras with a graph where the vertices represent the cameras, and the arcs represent the existence of overlaps. This means that there is an arc between two nodes if and only if the two cameras overlap. Based on this modelling, we observe that the group of cameras we have to find represent a set of nodes (cameras) that connect two by two. The latter is recognised by a *Clique* in graph theory. To find this type of graph (i.e., Clique), we have used the Bron-karboch Clique search algorithm, which allows us to find the maximum Clique, i.e., the maximum number of overlapping cameras.

The simulation results of these two proposed techniques show that the cameras are grouped with a stable rate of around 75% and 74% for the AHC-based method and the Clique-based method, respectively. Both methods have proven their efficiency concerning energy consumption by applying the coordination algorithm proposed by 4.4 compared to the conventional system. In this evaluation, we also observe that the Clique-based method is more efficient than the AHC-based method.

4.4 The proposed distributed collaborative multi-camera for tracking moving objects

Visual sensor networks play a crucial role in video surveillance systems based Internet of Video Things (IoVT) to detect, re-identify, recognise and track objects to protect people and sensitive places. Traditional video surveillance systems require more power, large bandwidth, and extra storage space during acquiring, processing, and transmitting multimedia data on the network. The system requirements increase due to the processing of the same events by several overlapping cameras. To address this problem, we propose a new distributed and collaborative tracking system based on the modern computing paradigm of IoVT. The system's main purpose is to increase network life, reduce processing costs, decrease communication data, and improve tracking quality. The proposed system operates in two steps: (*i*) Electing a leader among a set of cameras grouped according to the overlap degree of their FoVs, as we saw in the previous section. (*ii*) Choosing the best assistants from neighbouring cameras to maximise detection when the leader's vision is insufficient to track the object. Only the leader and their assistants are active. The other neighbouring cameras remain in an inactive state. To improve the real-time data processing, the system's load is distributed throughout the cloud-fog-mist computing architecture proposed in section 4.2.

4.4.1 Related works

4.4.1.1 IoVT paradigm based systems

In the last decade, the processes of VSS have become increasingly complex. In the smart city context, the number of cameras can be reached to many thousands of cameras, and the use of a single data centre requires significant computing and network resources to ensure the necessary quality of information. Formally, the adaptation of emerging technologies such as cloud, fog, and mist computing improves video surveillance systems' performance. These technologies provide a large capacity for real-time processing and storage. Despite all the advantages of these technologies, there are not many surveillance systems in the literature that adopt them [421].

- **Cloud computing based systems**

  After failing traditional systems to handle big multimedia data, cloud computing has successfully handled it [123, 426, 491]. The new cloud-based technology seems to have created and set a new trend in video surveillance systems.

  To control the extensive multimedia data generated by the video surveillance system, Xia Wei et al. [455] used cloud computing as a service platform to manage data cooperatively between system nodes. Another cloud-based system is proposed by [223]. The surveillance is performed through a multi-drone system called "DroneTrack", which uses a cloud robotics platform (Dronemap Planer) to manage and control drones on the Internet to track moving objects in real-time. The paper [474] presents a smart training room surveillance system based on mobile Internet and cloud computing. The cloud server is used to support the storage capacity and processing performance of the video transmission. Using a mobile device, the user can access the cloud server anytime and anywhere, either to obtain information on the status of surveillance rooms or to submit control commands. Mohammad et al. [23] propose a distributed surveillance system based on Edge and cloud computing to manage the distribution of bandwidth on cameras better and ensure efficient data transfer to the system's processing side.

- **Fog computing based systems**

  Due to the high latency of transferring massive multimedia data to cloud computing and the requirement of real-time processing in video surveillance systems, several researchers have proposed to customise the fog computing paradigm to address these challenges. With fog computing, the video surveillance data will be processed at the edge of the network. In other words, the intelligence will be pushed to the edge for extracting the relevant information and decide in real-time. The latter is recently demonstrated by [299], where the authors develop a fog computing infrastructure that uses deep learning models to process the video stream generated by surveillance cameras. The obtained experimental results confirm that using different deep learning models at different levels of the fog infrastructure allows processing the data stream in real-time and thus serve delay-sensitive applications.

  Anang et al. [25] propose an adaptation of fog computing to develop a distributed surveillance system for facial recognition. This system is divided into two phases: features extraction and recognition. The first phase is executed at the fog computing level to minimise massive multimedia data transmission. The second phase is performed in cloud computing due to the need for high computing power. Ning Chen et al. [99] propose

a traffic surveillance system based on fog computing. The system monitors vehicles on the roads in real-time through information fusion, rapid decision-making, and situational awareness. Juan et al. [435] propose a new infrastructure for smart city surveillance systems based on the Edge computing paradigm, allowing real-time process data.

4.4.1.2 Collaborative based systems

Recently, several video surveillance systems propose using multi-camera collaboration techniques to improve the quality of tracking moving objects and reduce the amount of redundant data caused by the overlap of several cameras' FoV. Jingjing et al. [164] propose a new architecture for a cooperative multi-UAV surveillance system based on the animal colony perception method for tracking moving objects. Santamaria et al. [361] present an intelligent, cooperative video surveillance system based on Edge and fog computing architectures to manage communication, distribution, and data processing of multi-cameras to detect and track objects. Luo et al. [259] propose a new algorithm for collaborative tracking of moving objects in a multi-camera network based on the layout of partially superimposed regions (LOPOR) to calibrate collectively several cameras, where each camera shares its information with the computing centre to obtain a global report on the objects.

The major drawback of all these proposals is that in a collaborative environment, the number of communications and the amount of information shared increases proportionally with the number of cameras, which requires additional bandwidth and energy. This increase may lead to an extra communication cost and risks of network overload and congestion due to the large number of messages transmitted on the network.

The possible solution to address these challenges is reducing or optimising the number of cameras participating in the surveillance process. Many solutions for optimising active cameras have been proposed in the literature on different networks, mainly wireless multimedia sensor networks (WMSN) and wireless visual sensor networks (WVSN). The first one contains scalar sensors (motion, temperature, light sensor, etc.) that transmit scalar data [483] and visual sensors that transmit images or videos—the second consists of many camera nodes [392].

In WMSN, Andrew Newell et al. [303] propose a new distributed trigger scheme to minimise active cameras. The authors use scalar sensors distributed in the surveillance zone to operate the smallest number of cameras. The scalar sensors detect events in the surveillance area and inform the neighbouring cameras. The system determines cameras that must be activated to capture and transmit the image based on the number of action messages received by the adjacent sensors located in the Field of view (FoV) of the concerned camera. In the same way, Salim and Ramad [357] propose a collaborative trigger scheme that allows the selection of a minimum number of cameras with better coverage and without loss of information by activating cameras located in the event area using neighbouring scalar sensors in addition to cameras that have FoV that intersect the event area. Priyadarshini et al. [335], propose to combine Centralized cum Sub-Centralised (CSC) scheme with a Scalar Intersection (SI) to manage multi-event coverage in WMSN. The target is to reduce the number of unnecessarily activated cameras to reduce power consumption, minimise redundant data, and maintain event regions' efficient discovery. They also designed a new algorithm to activate a minimum number of cameras with maximum area coverage. The latter is based on the election of a leader for each sensor sub-area to inform neighbouring cameras. In [336], the selected leader is the scalar sensor having the minimum average distance from the rest of the scalar sensors belonging to a particular sub-region. In the same context [337], the authors propose to select the scalar heads hierarchically, such that

the selected child node of a given parent node is placed at a distance equal to twice the depth of FoV of the cameras. These works focus a lot on energy minimisation and do not consider the usefulness of visual tracking. The cameras chosen based on the data received by scalar sensors are not necessarily the best in the tracking process. Also, in multi-event occurrences, the cameras are still active even if the environmental coverage rate decreases.

In WVSN based systems, Che-Cheng and Jichiang [90] propose collaborative coordination between all surveillance cameras using a single-leader election algorithm for automatically selecting the number of cameras that track objects. The latter does not optimise the number of active cameras but only determines the camera to follow an object. Also, the algorithm uses criteria and parameters that are not significant for the electoral process, such as the comparison between cameras identifier. To activate a set of cameras for tracking, Wei Li in [240] proposes calculating the correlation between the observations. In this case, the system activates only cameras with a high observation correlation between them. This method does not consider the cost of the energy consumed, and according to [500], this method cannot be applied to deterministic deployment.

In [284], the cooperative cameras are divided into a subgroup of active cameras and a subgroup of static cameras. The static cameras capture and integrate images to detect and track objects; meanwhile, the active cameras high-resolution images to track objects. This work focuses only on monitoring efficiency but does not consider energy consumption and storage space requirements. SanMiguel and Cavallaro [360] present a method for creating camera coalitions in multi-camera networks and demonstrate collaborative target tracking. The coalition formation was considered as a decentralised resource allocation process. In the process, the best cameras among those viewing are selected as a coalition using marginal utility theory. However, these approaches do not account for the dynamics of the network and the complexity level that will be increased to improve the continuous tracking of multi-camera objects.

4.4.1.3 Clustering based systems

Many studies use clustering methods to alleviate network traffic. Therefore, communication is allowed only between the cameras of a group containing a detected object. Yoder et al. [489] propose a facial tracking system distributed in the WVSN that uses a clustering protocol, with a leader camera in each cluster. The leader is selected according to the best distance between the location of the object's face and the camera. However, the distance does not allow choosing the best leading camera since the latter may not have the best vision on the object. Furthermore, all the cluster cameras participate in the tracking process, even if it is not necessary.

To improve the real-time tracking, Shuai Zhao et al. [510] propose a dynamic camera collaboration framework based on the community (or cluster) concept to detect overlapping areas. This mechanism allows through the Clique Percolation Method (CPM) to optimise the number of active groups. A new selection method proposed by Bhuvana et al. [60] allows cameras to make an independent decision on participation in the monitoring process. In this method, the selected cameras form a cluster that has a fusion centre between them. The role of this latter is the distribution of the global state to the cameras of the cluster. However, this method does not change the fusion centre and requires the knowledge of the total number of camera nodes that visualise the current target. To limit the communication area, avoid network congestion, reduce the redundancy of detected data and limit the rapid reduction in energy, Masoud Zarifneshat et al. [495] propose a distributed semi-localized clustering scheme with a dynamic selection of the

leader camera. However, the cluster size is static, which is not always effective, and the energy of the cluster leader rapidly decreases [9].

4.4.1.4 Probabilistic based systems

Probabilistic methods have also taken a very important part in the improvement of CCTV based IoT. Tessen et al. [411] propose a method to select a subset of cameras dynamically in a network to obtain the best possible tracking quality. The method is based on a well-known probabilistic mathematical theory, namely the Dempster-Shafer Theory of Evidence. Nevertheless, this selection process does not depend on the true observations [60]. Liang Liu et al. [250] addressed the problem of selecting the minimum number of cameras in WVSN based on the minimum cost criterion instead of the maximum utility. The proposed method consists of two phases: the target detection phase, which uses the Probing Environment and Adaptive Sleeping algorithm (PEAS) to select the most appropriate cameras, and the cooperative localisation phase based on Bayesian estimation. According to [448], this solution is not applicable to tracking objects with multi-cameras in a collaborative way.

Table 4.1 summarises the various conducted research between 2010 and 2020 in the field of video surveillance systems.

4.4.2 The proposed distributed collaborative tracking system

In this section, the proposed algorithm for distributed collaborative tracking objects is presented as the following.

4.4.2.1 System operation diagram

The operating scenario of the proposed system is very simple, as shown in Figure 4.21. At the first and after installing the cameras in the surveillance zone, each camera transmits to the responsible fog node all their information (coordinates, FoV, viewable angle θ and visible distance Rs). The next step is grouping cameras using one of the proposed methods discussed in section 4.3. After extracting the clusters of neighbouring cameras, the fog node informs each camera about its neighbours. Once a camera detected an object, the election process starts between them (neighbouring cameras only) to choose the leader. If the leader's vision is not sufficient to follow the object, this latter starts the process of selecting the assistant cameras. The assistant cameras are chosen among those maximising the leader's vision. After that, all concerned cameras start the tracking of the detected object. During the tracking process, the feature vector of the tracked object may change. In this case, the cameras that detect this change inform the responsible fog to update this object's feature vectors. At the end of each tracking process, the object will leave the FoV of one of the cameras following it. If the camera is an assistant, the system will continue to operate; but two schemes can arise if the camera is the leader. The first one is when the leader works on the tracking alone; in this case, the leader informs the fog node that the object is lost and send the last information (location and feature vector) of this object before leaving. In the second one, the leader works with a subset of assistant cameras, and in this case, the leader informs its neighbours to initiate the election process to choose a new leader.

TABLE 4.1: Summary of related work on CCTV

Refs	Infrastructure				System			Purpose	
	IoT-N	CC	FC	SCam	DA	CCam	OP	TO	OCam
[474]	✓	✓	✓	×	✓	×	×	×	×
[299]	×	×	✓	×	✓	×	×	×	×
[90]	×	×	×	×	✓	✓	-	✓	×
[510]	×	×	×	×	✓	✓	✓	✓	×
[23]	✓	✓	×	×	✓	×	✓	×	×
[455]	✓	✓	✓	×	-	×	✓	✓	×
[223]	✓	✓	×	×	✓	×	✓	✓	×
[25]	✓	✓	✓	×	✓	×	-	×	×
[99]	✓	✓	✓	×	-	×	✓	✓	×
[435]	✓	✓	✓	×	✓	×	✓	✓	×
[164]	✓	✓	×	×	✓	✓	✓	✓	×
[361]	✓	✓	✓	✓	✓	✓	✓	✓	×
[259]	×	×	×	×	×	✓	-	✓	×
[303]	×	×	×	×	✓	✓	-	×	✓
[357]	×	×	×	×	✓	✓	-	×	✓
[335]	×	×	×	×	×	×	×	×	✓
[336]	×	×	×	×	×	✓	×	×	✓
[337]	×	×	×	×	✓	✓	-	×	✓
[240]	×	×	×	×	✓	✓	✓	✓	✓
[489]	×	×	×	×	✓	✓	✓	✓	×
[60]	×	×	×	✓	-	-	-	✓	✓
[495]	×	×	×	×	✓	✓	-	✓	-
[411]	×	×	×	✓	✓	×	-	✓	✓
[250]	×	×	×	×	-	✓	-	✓	✓

(✓): Fully supported, (×): Not supported, (-): Not mentioned
IoT-N: IoT network, **CC**: Cloud computing, **FC**: Fog computing, **SCam**: Smart cameras, **DA**: Distributed architecture, **CCam**: Collaborative cameras, **OP**: Online processing, **TO**: Tracking objects, **OCam**: Optimisation cameras.

4.4.3 Camera collaboration algorithm

In the following, we introduce a new camera collaboration algorithm based on the general diagram presented in the section 4.4.2.1. The latter is characterised by an almost negligible message exchange rate and consists of three main steps: leader election, assistant selecting and collaborative tracking. The first step is the lead camera's election, which has the best vision on the detected object. In addition to the monitoring task, the leader's role is to select the assistant cameras and coordinate the information exchange between them. The elected leader performs the second step to search for assistant cameras and select -if needed- the most appropriate for real-time tracking, minimum redundant data, lower storage space, and less energy consumption with better tracking quality. The third step is the collaborative tracking of the object by the leader and their assistants.

Before starting the discussion of the proposed algorithm and to better follow up, the used process, variables, functions and message types used in the algorithm with the role of each one are respectively summarised in the Tables 4.2, 4.3, and 4.4.

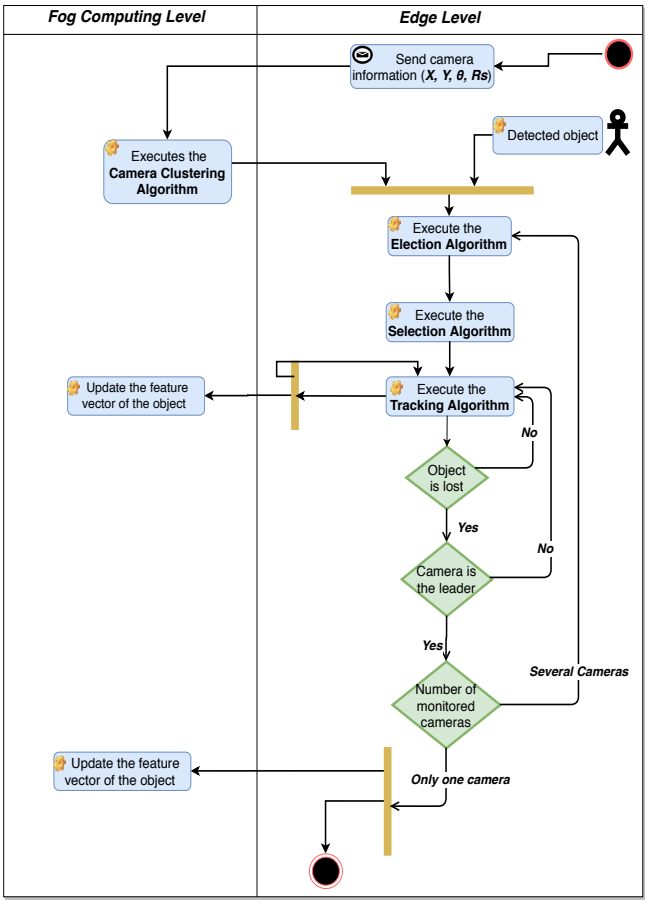


FIGURE 4.21: Conceptual diagram of the proposed algorithm

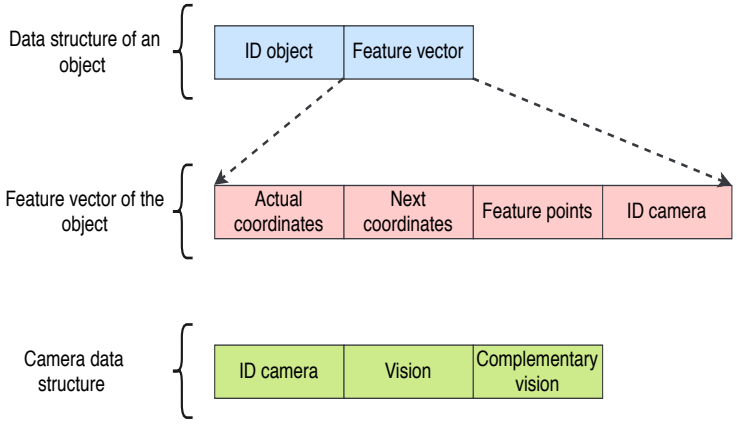


FIGURE 4.22: Data structures

To give more detail on the collaboration and coordination algorithms, we define in the following, the behaviour of the cameras and the related events.

- **Cameras can detect multiple objects, but they track only one:** this rule reduces the complexity of the tracking process (detection, feature extraction and tracking process). Therefore, if a camera is tracking an object and detects another object, two cases arise. If the camera is a leader, it remains focused on the object being monitored; otherwise, the camera interrupts the tracking and starts the newly detected object's election process.

TABLE 4.2: Description of the used variables

Variables	Description
obj	Data structure of an object (see Figure 4.22)
id_o	Object identifier
vx	Feature vector extracted by the camera
cam	Camera data structure (see Figure 4.22)
assistants	List of camera c_i's assistants
id_c	Camera identifier
vs	Camera vision on the object
$\overline{vs}$	Complementary vision
max_vs	Variable contains the maximum vision
leader	Variable contains the identifier of the leader camera
$\overline{leader}$	Variable contains the ID of the camera that will probably become the leader
state	State of the camera ∈ {"free", "candidate", "sleep", "active"}
threshold	The necessary vision for tracking the object
nb_msg	Number of messages received by the camera
sum_comp_vs	The sum of complementary visions
cur_cord	The current coordinates of the object being tracked
nex_cord	The next coordinates of the object being tracked

TABLE 4.3: Description of the functions

Functions	Description
get_vx()	Calculate the feature vector of the detected object
get_vs(vx)	Calculate the camera vision on the detected object from their feature vector vx
get_com_vs(vx_i, vx_j)	Calculate the complementary vision of the camera c_i with regard to the camera c_j which have the vision vector vx_i and vx_j respectively
max_vs_com($L[]$)	A function that returns the camera having a maximum complementary vision
create_id_object()	A function that works on the generation of new identifiers for the new objects

This constraint ensures that the first object is followed up by at least the head camera and ensures that the newly detected object has at least one candidate camera.

- **Each camera participates in only one election at a time:** if there is more than one election operation, the cameras participate in only one of them. This rule avoids a conflict that occurs, like when a camera participates in two election operations, and it is chosen as a leader in both.
- **The leader camera is the camera that has the max vision on the detected object:** the leader always has the maximum vision. This constraint allows minimising the number of assistant cameras will choose by the leader to track the object.
- **The leader's election is carried out only once for each object which is already being tracked:** if a new camera detected an object that is already being tracked by a set of cameras, the latter would be blocked for tracking this object.

TABLE 4.4: Description of messages

Messages	Description
Election (...)	Sent by the camera that detected an object to the neighbouring cameras. It informs that there is a detected object in order to start the election operation between the cameras that detected or tracked that object
Accept (...)	Confirmation message sent by the leader camera to inform the candidate cameras that they have been accepted for tracking the object with it
Reject (...)	Sent by the leader camera to inform the candidate cameras that they have been refused
Re-election (...)	Sent by the leader camera to indicate that the tracked object is lost (leaving their FoV)
Leave (...)	Sent by the assistant cameras to the leader when the tracked object is lost or when another object has been detected in their FoV
Request_Id_Object (...)	Sent to the fog node to request an ID for the detected object
Inform_Id_Object (...)	Sent by the fog to the camera that requested an ID, the answer contains the object identifier detected by this camera
Lost_Object (...)	Sent by the leader camera to inform the fog that the tracked object is lost and there are no assistants
Inform_Fog (...)	Sent by the fog to the neighbouring fog nodes when an object leaves its coverage area

- **The camera leaving the group informs the leader:** this constraint allows the leader to remove the camera from the set of assistants and guarantees the optimisation of communications between the cameras and avoid network congestion.
- **The leader stop assistants election when the accrued visions reached the threshold:** in the step of assistant cameras selection, the leader stops selecting when the sum of its vision with the selected assistant cameras' vision is greater than or equal to the sufficient vision for tracking the object (*threshold*). The maximum vision must be reached with the minimum of cameras.
- **The leader informs the corresponding fog node when the object disappears from its FoV:** The object can be lost in internal or external dead zones (see Figure 4.23). In this case, the leader informs the fog node that the object is lost. If the object is lost in the external dead zone, the fog sends an information message to the neighbouring fog nodes that will probably receive the object.

In the proposed system, the camera can be in four states during its life cycle (Figure 4.24):

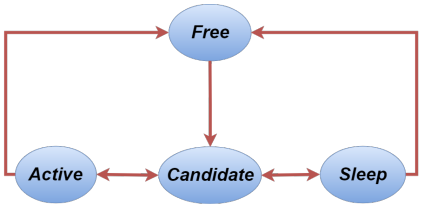


FIGURE 4.24: State transition diagram of the camera in its life cycle

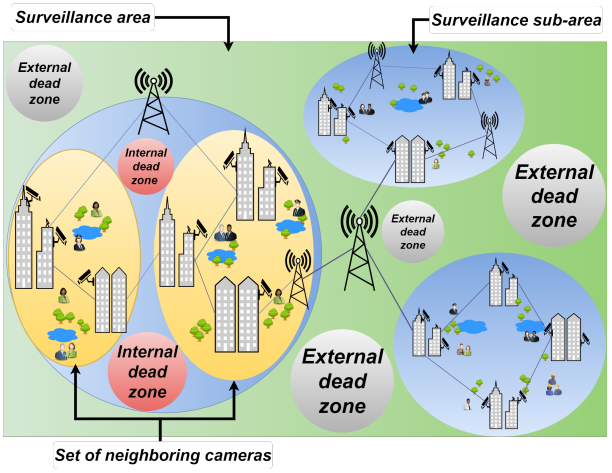


FIGURE 4.23: Some examples of dead zones

- **A Free camera** is a camera without any moving objects in its FoV.
- **A Candidate camera** is a camera that has detected an object and ready for the election process.
- **A Sleeping camera** is a camera that has detected an object which is currently tracked by another camera.
- **An Active camera** is a camera in the process of tracking an object, whether it's a leader or an assistant.

The proposed approach consists of a set of algorithms executed by the cameras or by the fog computing nodes according to the requests received in messages or detected events as shown in the Table 4.5 (the numbering of algorithms does not mean the scheduling of the execution). The rest of this section describes the proposed algorithms starting with those executed at the edge or camera level, then the algorithms executed in the fog nodes.

4.4.3.1 At the camera level

Algorithme 4.5 : Election preparation

1: $\overline{leader} \leftarrow -1$;
2: $\overline{leader} \leftarrow id\_c$;
3: $state \leftarrow$ "$candidate$";
4: $nb\_msg \leftarrow 0$;
5: $candidate\_list \leftarrow \varnothing$;
6: $rejected\_candidate\_list \leftarrow \varnothing$;
7: $assistants \leftarrow \varnothing$;
8: $vx \leftarrow$ get_vx ();
9: $vs \leftarrow$ get_vs (vx);
10: sum_comp_vs $\leftarrow vs$;
11: $max\_vs \leftarrow vs$;

The objective of algorithm 4.5 is to avoid code redundancy. It is a set of instructions to be executed before starting the election to prepare each camera's local variables.

TABLE 4.5: Execution levels of each algorithm

Algorithms	Free camera	Candidate camera	Sleep camera	Active camera		Fog node
				Assistant camera	Leader camera	
Algorithm 4.5	✓	✓	✓	✓	✓	
Algorithm 4.6	✓		✓	✓		
Algorithm 4.7		✓				
Algorithm 4.8		✓				
Algorithm 4.9					✓	
Algorithm 4.10		✓				
Algorithm 4.11		✓				
Algorithm 4.12				✓	✓	
Algorithm 4.13			✓	✓		
Algorithm 4.14					✓	
Algorithm 4.15						✓
Algorithm 4.16						✓
Algorithm 4.17						✓
Algorithm 4.18						✓

Algorithme 4.6 Upon **: detecting an object**

1: **if** ($state =$ "$free$") **or** ($state =$ "$sleep$") **or** ($state =$ "$active$" **and** $id\_c \neq leader$) **then**
2: **** start video recording ****
3: **if** ($state = active$) **then**
4: Send **Leave** (id_c) to $leader$;
5: **end if**
6: **** execute algorithm 4.5 ****
7: Send **Request_Id_Object** (vx) to fog;
8: **end if**

Algorithm 4.6 is executed by cameras that detect an object in their FoV. In the first step, the camera tests if its current state allows it to start an election process. If the constraints are verified, the camera starts video recording. Suppose the camera is an assistant to another leader. In that case, it sends a resignation message "*Leave*" to its leader, executes the election preparation algorithm and then sends a message to the fog node corresponding to the surveillance zone to get the detected object identifier.

When the camera receives a response from the fog on the requested identifier (algorithm 4.7), the latter becomes a local object identifier, and an "*Election*" message is sent to all neighbours.

Algorithme 4.7 : Upon receiving **Inform_Id_Object** (id_o_j) from fog

1: $id\_o \leftarrow id\_o_j$;
2: Send **Election** (id_o, vx, vs) to neighbours;

Algorithme 4.8 : Upon receiving **Election** (id_o_j, vx_j, vs_j) from cam_j

1: **if** (($state \neq$ "$sleep$") **then**
2: **if** (($state \neq$ "$free$") **and** $id\_o = id\_o_j$) **then**
3: **if** ($state =$ "$active$") **then**
4: **if** ($leader = id\_c$) **then**
5: Send **Reject** ( ) to cam_j;
6: **end if**
7: nb_msg - -;
8: **else if** ($max\_vs < vs_j$) **or** (($max\_vs = vs_j$) **and** ($\overline{leader} < cam_j$^**.**$id\_c$)) **then**
9: $max\_vs \leftarrow vs_j$;
10: $\overline{leader} \leftarrow cam_j$^**.**$id\_c$;
11: **else if** ($\overline{leader} = id\_c$) **then**
12: $\overline{vs} \leftarrow$ get_com_vs(vx, vx_j);
13: **if** (($\overline{vs} \neq 0$) **and** ($vs_j \neq 0$)) **then**
14: $candidate\_list$[].add((c_j, $\overline{vs}$));
15: **else**
16: $rejected\_candidate\_list$[].add((c_j, vs_j));
17: **end if**
18: **end if**
19: $nb\_msg + +$;
20: **if** ($nb\_msg = \#neighbours$) and ($state \neq$ "$active$") **then**
21: **if** ($\overline{leader} = id\_c$) **then**
22: $leader \leftarrow id\_c$;
23: $state \leftarrow$ "$active$";
24: **** started tracking process and execute algorithm 4.9 ****
25: **else**
26: $leader \leftarrow \overline{leader}$;
27: $candidate\_list \leftarrow \varnothing$;
28: $rejected\_candidate\_list \leftarrow \varnothing$;
29: **end if**
30: **end if**
31: **else**
32: Send **Election** (id_c, $\varnothing$, 0) to c_j;
33: **end if**
34: **else if** $vs_j \neq 0$ **then**
35: Send **Election** (id_c, $\varnothing$, 0) to c_j;
36: **end if**

Algorithm 4.8 is executed when a camera receives the message "Election" from neighbouring cameras. If the latter is not free and the received object is the same currently tracked, the leader camera sends a message "*Reject*" to the transmitting camera to inform it that the object is already tracked. Suppose it's not the case. If the transmitting camera's vision is greater than the maximum vision max_vs, the transmitting camera is set as a potential leader. In the case of equality, the camera with the highest identifier is chosen as a potential leader. Suppose the receiving camera is the potential leader, and the complementary vision between the two vision vectors is not equal to zero. In that case, the transmitting camera is added to the accepted candidate list. Otherwise, it is added to the rejected candidate list.

When the receiving camera is not free, or the received object is different from the detected

object, the camera responds with an "*Election*" message with a void vision to be excluded later. The same treatment is applied to the sleeping cameras.

All participating cameras identify the end of the election when they receive a message from all their neighbours. If the camera remains the potential leader, it becomes the leader, starts the tracking process and goes to the assistant selecting step. Otherwise, it saves the potential leader's identifier as the leader's identifier and clears the lists' content.

Algorithme 4.9 : Selection

```
1: k ← 0;
2: while (k < |candidate_list| and sum_comp_vs < threshold) do
3:     candidate ← max_vs_com(candidate_lis[]);
4:     sum_comp_vs ← sum_comp_vs + candidate^.vs;
5:     Send Accept () to candidate^.id_c;
6:     assistant[].add(candidate);
7:     k + +;
8: end while
9: if (k < |candidate_list|) then
10:    while (k < |candidate_list|) do
11:        Send Reject () to candidate_list[k]^.id_c;
12:        k + +;
13:    end while
14: end if
15: for (C ∈ rejected_candidate_list[]) do
16:    if (C^.vs ≠ 0) then
17:        Send Reject () to C^.id_c;
18:    end if
19: end for
```

The assistant camera selection algorithm (algorithm 4.9) allows the leader to choose the minimum number of assistants with a maximum complementary vision to track the object. To achieve such a result, the leader scrolls through the list of candidates (*candidate_list*[]) and takes the camera having the maximum complementary vision to its vision. The leader informs each selected camera through the message "*Accept*". This step ends when the sum of visions has reached the maximum threshold (according to the fifth constraint); otherwise, all the neighbouring cameras are selected. The leader informs the remainder cameras, if they are present, with a "*Reject*" message.

According to the algorithm 4.9, the candidate camera receives one of two messages from the leader. Either a confirmation message ("*Accept*") or a nonacceptance message ("*Reject*"). Upon receipt of the "*Accept*" message, the camera changes its state to "*active*" and starts the tracking process (algorithm 4.10). When a candidate camera receives a "Reject" message, the camera interrupts the video recording, changes its state to "*sleep*" and adds the detected object to the list of blocked objects (algorithm 4.11).

Algorithme 4.10 : Upon receiving **Accept** (id_j) from c_j

```
1: state ← "active";
2: leader ← −1;
3: ** The Tracking Process Started **
```

Algorithme 4.11 : Upon receiving **Reject ()** from c_j

```
** Stop Video Recording **
state ← "sleep";
blocked_object_list.add(id_o, vx);
```

At any time, an active camera may lose the tracked object when this latter leaves its FoV (algorithm 4.12). In this case, the camera stops recording the video and stops the tracking process. If the camera is a leader and has no assistants, it sends the message "*Lost_Object*" to inform the fog that the object is lost; otherwise, it sends the message "$Re - election$" to all neighbours. When an assistant camera loses the tracked object, it sends a resignation message "*Leave*" to the leader and initialises its state to "$free$". In some cases, the camera loses one of the detected objects and not the tracked one. In this case, the camera removes it from the list of blocked objects "*blocked_Object_List*".

Algorithme 4.12 : End of the Event

```
if tracked object is lost then
    ** stop video recording and tracking process **
    if (leader = id_c) then
        if (#assistant = 0 ) then
            Send Lost_Object (id_o, vx) to fog;
        end if
        Send Re-election (id_c, id_o) to neighbours;
    else
        Send Leave () to leader;
    end if
    state ← "free";
else
    blocked_object_list.remove(the lost object);
end if
```

When the leader loses the tracked object, it sends a "$Re - election$" message for all cameras. Indeed, some cameras can get the tracked object in their FoV, but the latter is inserted in the "*blocked_object_list*[]". The loss of the object to the leader gives a chance to all cameras that having seen this object to follow it. In this situation, two cases may arise (algorithm 4.13): the receiving camera is an assistant, or it is a camera in "*sleeping*" state, having the object in its "*blocked_object_list*[]" as well as in its FoV. In both cases, the camera executes the election preparation algorithm to select a new leader and then send the neighbours' election message. The object will be removed from the "*blocked_object_list*[]" if the camera is not an assistant.

The leader automatically removes the resigning camera from its assistant cameras list when it receives the "*Leave*" message from one of its assistants (algorithm 4.14).

4.4.3.2 At the fog computing level

Since the cameras are grouped in clusters with separate fog nodes, tracking an object between the fog regions requires synchronisation between them. The fog intervention ensures tracking continuity when an object shifting occurs from one zone to another and reduces ID-switch errors. To do this, as soon as a camera detects an object, it requests an identifier from the fog

Algorithme 4.13 : Upon receiving **Re-election** (id_c_j, id_o_j) from c_j

```
1: if (leader = id_c_j) then
2:     ** execute algorithm 4.5 **
3:     Send Election (id_o, vx, vs) to neighbours;
4:     nb_msg ++ ;
5: else if (state = sleep and id_o_j ∈ blocked_object_list[]) then
6:     Object ← blocked_object_list[].get(id_o_j);
7:     if (Object stay in FoV) then
8:         ** execute algorithm 4.5 **
9:         Send Election (id_o, vx, vs) to neighbours;
10:        nb_msg ++ ;
11:    end if
12:    blocked_object_list[].remove(id_o_j);
13: end if
```

Algorithme 4.14 : Upon receiving **Leave**(id_c_j) from c_j

```
1: assistant[].remove(id_c_j);
```

Algorithme 4.15 : Upon receiving **Request_Id_Object**(vx_j) from c_j

```
1: if ( vx_j^.cur_cord ∈ list_objs_cur_tracking) then
2:     obj ← list_objs_cur_tracking.get(vx_j^.cur_cord);
3:     id_o ← obj^.id_o;
4: else if (vx_j^.nex_cord ∈ list_received_obj then
5:     obj ← list_received_obj.get(vx_j^.nex_cord);
6:     id_o ← obj^.id_o;
7: else if (Vector_j^.feature_points ∈ list_his_obj then
8:     obj ← list_his_obj.get(vx_j^.feature_points);
9:     id_o ← obj^.id_o;
10: else
11:    id_o ← create_id_object();
12:    obj ← new_object ();
13:    obj^.id_o ← id_o;
14: end if
15: Send Informer_Id_Object (id_o) to c_i;
16: obj.add(vx_j);
17: list_objs_cur_tracking.add(obj);
```

(algorithm 4.15). The latter uses the feature vector received from the transmitting camera and performs searching or matching with the objects already tracked or being tracked. If the match is successful, then the fog sends the recovered identifier to the transmitting camera; otherwise, it creates a new identifier for this object. The fog can also receive messages from the nearest fog nodes in the case of loss of the tracked object. In this type of scenario, the fog uses the last recorded coordinates of the lost object in addition to the feature vector to check if it is in its area (algorithm 4.16).

Upon receipt of the "$Inform\_Fog$" message (algorithm 4.17) from the fog nodes, the receiving fog adds the received object to the list of received objects *list_received_obj* which means that

Algorithme 4.16 : Upon receiving **Lost_Object** (id_o_j, vx_j) from c_j

```
1: if (vx_j^.cur_cord ∉ fog zone) then
2:     obj ← list_objs_cur_tracking.get(id_o_j);
3:     obj.add(vx_j);
4:     Send Inform_Fog (obj) to fog nearest to cur_cord;
5:     list_his_obj.add(obj);
6: end if
```

Algorithme 4.17 : Upon receiving **Inform_Fog** (obj_j) from fog_j

```
1: list_received_obj.add(obj_j);
```

this object can enter their area. This type of messages allows minimising the number of identity switches when the object travels through the external dead zone.

The object's feature vector may change over tracking. When the camera detects a change, the new feature vector is sent to the fog in the message "$Vector\_Update$". Upon receipt of this message, the fog stores the current object vector to the ($list\_his\_obj$) list and replaces it, in the list of current tracked objects ($list\_objs\_cur\_tracking$), with the new one(algorithm 4.18).

Algorithme 4.18 : Upon receiving **Vector_Update** (obj_j) from c_j

```
1: obj ← list_objs_cur_tracking.get(obj_j^.id_o);
2: list_objs_cur_tracking.remove(obj_j^.id_o);
3: list_objs_cur_tracking.add(obj_j);
4: list_his_obj.add(obj);
```

4.4.3.3 Case study

To better show how the proposed algorithms work, we present the system's behaviour about some scenarios graphically (Figure 4.25 and 4.26). The used environment consists of five cameras separated into two groups. The first group contains C_1, C_2 and C_3 and the second group contains C_4 and C_5. The maximum vision threshold is set to 80%.

Figure 4.25-(a) shows the initial state's surveillance zone, where all cameras are free ($state$ = "$free$"). In Figure 4.25-(b), the first object has entered into the FoV of the cameras C_1 and C_2. According to the proposed algorithm, C_1 and C_2 have the right to launch the election operation since they are in the free state (algorithm 4.6). Each camera starts recording the video on this object and then initialises their variables according to algorithm 4.5. The vision of C_1 and C_2 is respectively set to $vs = 60\%$ and $vs = 40\%$. After completing algorithm 4.5, the cameras send a message $Requeste\_Id\_Objet$ to the fog to find out the object identifier. The fog, in turn, executes algorithm 4.15 to search for this object in the object lists (current list of tracked objects, received objects list and historical objects list). Assuming that this object does not exist in all lists (i.e., never tracked in this area), the fog creates a new identifier for the object (for example "$id\_o = O_1$") and sores each vector received from C_1 and C_2. The created identifier O_1, is sent to C_1 and C_2.

Upon receipt of the fog's response (algorithm 4.7), C_1 and C_2 send an election message to all neighbours and execute algorithm 4.8 for each received election message. Figure 4.26-(a) shows the exchange of messages between neighbouring cameras, according to the election algorithm.

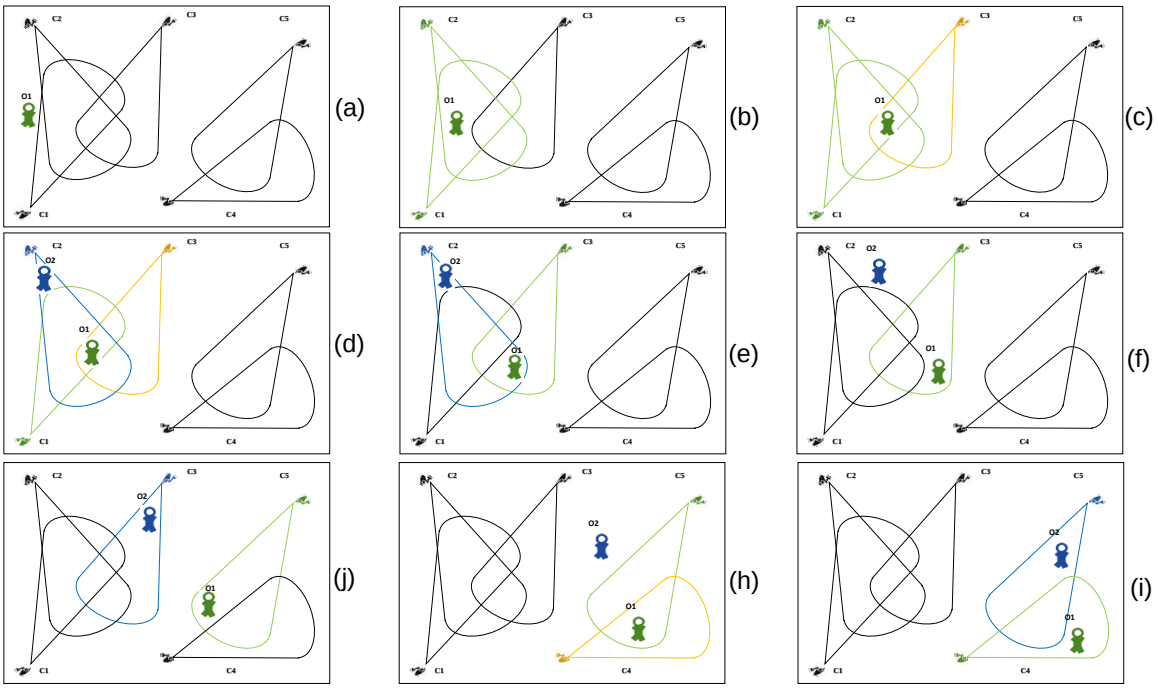


FIGURE 4.25: An example showing the functioning of the proposed algorithms in an artificial context

At the end of the election, each candidate camera knows the leader camera, for example C_1 and C_2 know that C_1 is the leader through the variable probable leader which contains the identifier of C_1 ($\overline{leader} = C_1$). The leader camera C_1 puts its identifier in the leader variable changes its state to active "$state := active$" and starts the tracking process.

After the election, C_1 executes the selection algorithm (algorithm 4.9) since its vision on the object O_1 is lower than the threshold. C_1 needs assistant cameras for good tracking. The execution of the algorithm 4.9, leads O_1 to choose C_2 as assistant camera (see the last message sent by C_1 to C_2 in Figure 4.26-(a)). C_2 executes the algorithm 4.10 after receiving the confirmation of C_1, puts the identifier of C_1 in the *leader* variable, changes its state to active ("$state = active$") and starts the tracking process with C_1. Figure 4.25-(b) shows that C_1 and C_2 are currently tracking the object O_1. The state of the other cameras remains in the sleep mode, since they have not seen the object ($vs = 0\%$).

During tracking, if any change occurs in the object's features vector, the camera sends an update message to the fog containing the new vector. The fog adds the received vector to the list of vectors of the corresponding object (algorithm 4.18). This option allows grouping different features vectors of the same object from different angles and places to create a complete profile of the object. This latter will be used to detect any trace of the concerned object in the surveillance area.

Over time, the object O_1 moves within C_3 FoV (Figure 4.25-(c)). The latter detects O_1 with a vision of 50%, starts video recording and executes algorithm 4.5 then sends a message to the fog to get back the O_1 identifier. Upon receipt of the identifier (algorithm 4.7), the camera sends the election message to its neighbours C_1, C_2. Knowing that $C1$ and C_2 are two active cameras, the leader C_1 sends a *Reject* message to say that this object (O_1) is already tracked, as shown in Figure 4.26-(b). In this case, C_3 will be blocked ($state =$ "$sleep$") according to the

FIGURE 4.26: Communication diagram between cameras for collaborative tracking

algorithm 4.11. C_2 as an assistant camera, simply ignore the message (algorithm 4.8) since the detected object is already tracked by C_2.

In the next scenario, another object enters into the FoV of C_2 with a vision rate of $vs = 70\%$ (Figure 4.25-(d)). At this moment, C_2 is tracking the O_1 with the leader C_1, but according to algorithm 4.6, C_2 is authorised to launch the election process since it is an assistant camera. For that, C_2 sends a resignation message to the leader, executes algorithm 4.5 then sends a *Request_Id_Object* message to the fog to get back the identifier of this object. Assuming that the detected object does not exist in the fog's object lists, the latter executes algorithm 4.15, creates a new identifier O_2 for this object and sends it to C_2. After the receipt of the object's ID, C_2 sends an election message to the neighbours, as shown in Figure 4.26-(c). C_2 becomes the leader for O_2 as it is the only camera that can see O_2 among all neighbours cameras.

As shown in Figure 4.25-(e), the object O_1 leaves the FoV of the leader camera C_1 (algorithm 4.12). In this case, it sends a *Lost_Object* message to the fog computing because it is the only tracker of O_1 and a *Re_election* message to the neighbours C_2, C_3 for possible re-election. Knowing that C_3 is in the sleep state, O_1 has been already detected by C_3, and it is still in its FoV; it executes algorithm 4.5 to prepare for the election operation and sends the *election* message to all neighbours except C_1. C_2 is a leader camera that tracks O_2, so it does not participate in the elections. Figure 4.26-(d) further clarifies this scenario.

When O_2 leaves the FoV of C_2 (Figure 4.25-(f)), the latter sends the message *Lost_Object* to the fog. In turn, the fog predicts the next position of the leaving object and verifies whether if it is still in its area (algorithm 4.16). In our scenario, the object remains in the surveillance zone, so the fog does nothing. C_2 also sends a $Re-selection$ message to the neighbours; however, C_1 and C_3 do not see the object. C_3 will behave in the same way as C_2 when the object O_1 leaves its FoV.

Over time, the C_5 camera detects O_1 and C_3 detects O_2 (Figure 4.25-(j)), and each of them follows the same steps. They start video recording and send the message *Request_Id_Object* to the fog. Upon receipt of the answer, they send the election messages to their neighbours. At the end of the process, C_3 becomes the leader for O_2, and C_5 becomes the leader for O_1, and they start tracking the detected objects. There is no assistant in this scenario because the neighbours of C_3 do not see the object, same for the neighbours of C_5.

Figure 4.25-(h) shows the leaving of O_2 from the FoV of C_3 and the detection of O_1 by C_4. When C_3 detects the end of tracking of O_2, it informs the fog and then broadcasts the $re-election$ message to its neighbours. Based on the prediction of the next position, the fog does nothing because the object is still in its surveillance zone. The neighbouring cameras also do nothing since they do not see the object. The detection of O_1 push C_4 to start the election process with C_5 (algorithm 4.6, 4.5, 4.7). Since O_1 is already followed by C_5 as a leader, the latter responds to C_4 with a *Reject* message (algorithm 4.8), which put the state of C_4 to sleep mode as shown in the scenario in Figure 4.26-(e).

When the object O_1 leaves C_5 FoV (Figure 4.25-(i)), this latter sends the message *Lost_Object* to the fog and a *Re_election* message to C_4 knowing that C_5 is the only one that tracks O_1.

At the fog level, nothing is happening, since the object O_1 did not leave the surveillance area(algorithm 4.16). The camera C_4 firstly tests if the object was already detected (in the blocked object's list) and secondly if it is still in its area (algorithm 4.13). In the proposed scenario, both conditions are verified, so C_4 starts the election process, as shown in Figure 4.26-(f).

The Figure 4.25-(i) also shows the entry of the object O_2 in the C_5 FoV which becomes free after the leaving of O_1. C_5 starts the election process (algorithm 4.6, 4.5, 4.7) and becomes a leader on O_2 since C_4 is the leader for O_1 (see Figure 4.26-(f)).

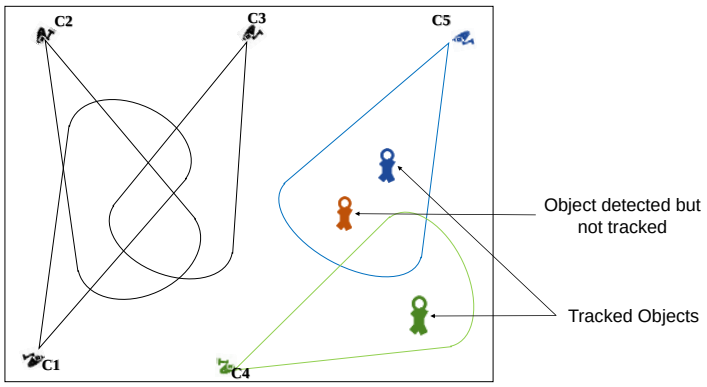


FIGURE 4.27: Weakness point

C_4 and C_5 continue to track O_1 and O_2 until they leave their FoV, then send a *Lost_Object* message to the fog and the $Re-election$ message to the neighbours (algorithm 4.12). Based on the prediction of the next positions, the fog knows that O_1 and O_2 have left its surveillance zone and, in turn, send the $Inform\_Fog$ message to inform neighbouring fog nodes of the possibility that these objects may have entered their zones (algorithm 4.16). All receiver fogs add the received objects in the received objects list "*list_received_obj*" (algorithm 4.17).

The proposed scenario does not allow a camera to track more than one object at a time. This constraint is due to the requirement of machines with more computing capacity and higher energy which is not present in our case. However, all algorithms can be easily used in a multi-tracking environment. Indeed, each camera can start the proposed algorithms separately for each object if its capacity can allow that. The limitation can only be at the camera level since fog computing's definition considers that its power and storage capacity is sufficient. Further, in video surveillance systems, the tracking is performed only for suspect objects and not for all moving objects, making such a scenario obsolete only when the number of suspicious objects is greater than the number of cameras (Figure 4.27).

4.4.4 Simulation and results

This section discusses the simulation results of the proposed cameras collaboration algorithm for tracking moving objects. The difficulty of having access to a real monitoring platform and the absence of a public simulator, which allows certain flexibility in communications, has led us to develop our own simulator[2]. This latter was implemented in JAVA with the JADE[3] platform, and each camera, object and fog node was modelled by an agent. The system was executed on a workstation equipped with an Intel(R) Xeon(R) E5-2620V3 2.4 GHz processor, 16 GB memory capacity, and operating system-based Linux. The simulation was performed by a network of 240 cameras distributed over 20 sub-areas, where each one includes 12 cameras and one fog node. The cameras in every sub-area are divided into two neighbouring camera groups, including 6 of the most overlapping cameras.

[2]The full JAVA code is available on GitHub platform: https://github.com/AlaEddinePHD/Collaborative-camera-simulator

[3]JADE: Java Agent Development Framework

- **Motivation to use the multi-agent paradigm for our simulator implementation**

  The major development that has enabled the design of truly autonomous, distributed artificial intelligence systems is the agent paradigm [383, 405, 458]. The agent paradigm extends the object-oriented paradigm [67, 184]. An object is the basic passive element commonly used in object-oriented design, extended to create autonomous software modules capable of independent reasoning and self-adaptation to an evolving external world called agents (agent-oriented design). Therefore, a multi-agent system (MAS) is a community of such autonomous, intelligent, and goal-oriented agents who cooperate and coordinate their decision-making to reach a global goal [34]. Despite the often heated nature of the debate about agent definitions, some characteristics are agreed upon [516]:

  – *Autonomy*: agents are partially independent, self-aware, autonomous and capable of exchanging knowledge about the environment. The metric to define the autonomy level is given in the article [109].

  – *Reactivity*: agents can perceive and adapt to environmental changes. These are key points of an agent's autonomy.

  – *Local views*: no agent has a full global view of the system, or the system is too complex for an agent to make practical use of such knowledge.

  – *Decentralization*: there is no designated controlling agent (or the system is effectively reduced to a monolithic system).

  To benefit from the characteristics of the multi-agent system, we use it to implement our simulator. Many agent platforms have been developed to simplify the development of multi-agent systems. For the demonstration of the proposed simulator, JADE is chosen as the agent development platform. JADE is widespread agent-oriented middleware with clearer design and implementation features, better documentation, and completely distributed middleware. In JADE platforms, the agents communicate with each other via the FIPA[4] communication protocol. Figure 4.28 shows the software architecture of the developed simulator.

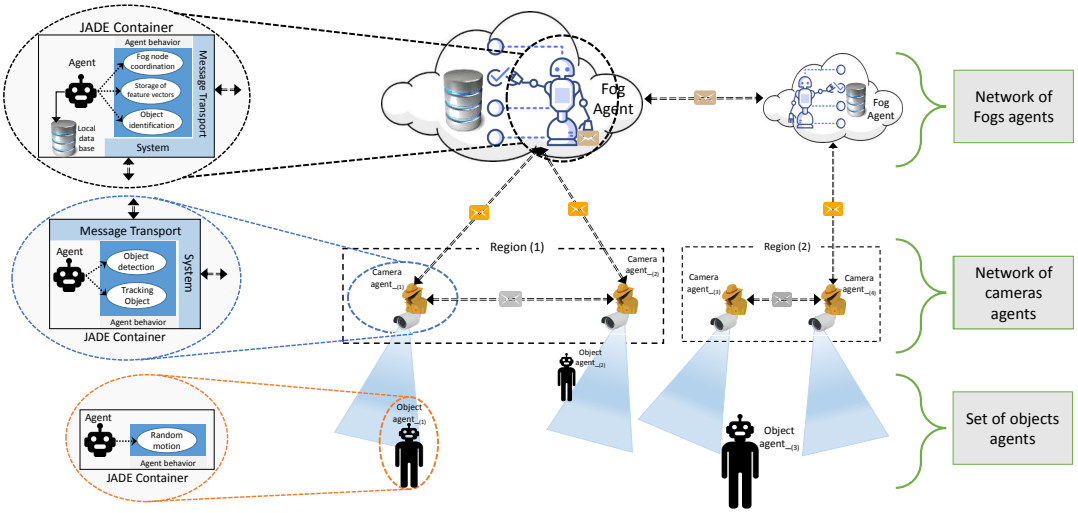


FIGURE 4.28: Software architecture of our developed simulator

[4]FIPA: Foundation for Intelligent Physical Agents.

The experiment was executed 10 times, using 500 objects in each one. The duration of one experiment takes, on average, from 6 to 7 hours. The iterations are distinguished by the number of objects present at the same time in the surveillance zone. To obtain the closest results to reality, each object enters the surveillance zone with random coordinates and moves freely with random trajectories.

Figure 4.29 shows two adjacent surveillance zones. The system coloured the camera's FoV with four colours to indicate the state of each of them: green FoV for leaders, orange for the assistant cameras, black to represent the free cameras, and red for the blocked cameras.

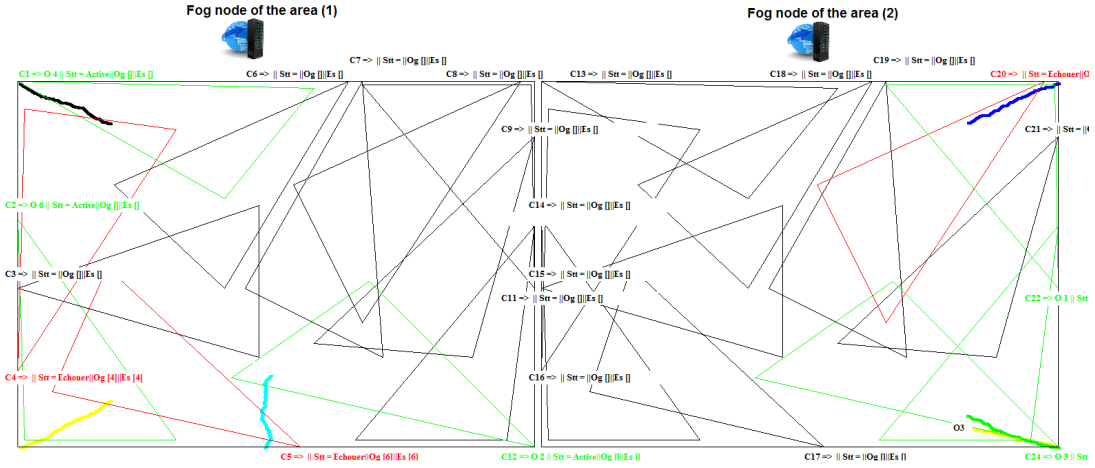


FIGURE 4.29: Screenshot of the developed simulator with two surveillance zones

Table 4.6 shows the system's number of actions on the cameras' state when objects pass through their FoV. The system's behaviour is studied according to the number of objects simultaneously in the surveillance zone. It is clear that the activation rate of cameras as a leader increases with the increase of the number of parallel objects in the surveillance area, while the number of assistants decreases significantly. This is due to multiple objects in the overlap area, leading the system to switch the cameras from the assistant state to the leader state. We also notice a decrease in the cameras' blocking actions with the increase in the number of objects in parallel to meet the system's needs.

To evaluate our algorithm in terms of energy consumption, bandwidth and amount of the produced data, we calculated the time taken by active and passive cameras. Figure 4.30 shows that the proposed system allows an average saving time of 22.62 % of the total execution time. We note that the saving time decreases slightly with the increase of the number of parallel objects in the surveillance area. The reduction in camera operating time leads to decreased energy consumption, bandwidth, and amount of the produced data.

In addition, we have calculated some quality measurement widely used in state of the art to evaluate tracking in video surveillance systems. Table 4.7 provides a short description of the evaluation criteria, and Figure 4.31, 4.32 and 4.33 show the proposed system's behaviour related to these latter.

The system gave good results even when the number of objects in parallel increases in the surveillance zone. In Figure 4.31, the system had a slight increase in FA due to the recognition module, which failed to detect the moving object correctly. In fact, surveillance cameras do not have the same fields of view, and the characteristics of an object facing the camera are not the same when the same object is positioned in the rear or profile.

TABLE 4.6: Number and types of activations of the cameras during the tracking of 500 objects

# of parallel objects	1	2	3	4	5	6	7	8	9	10
# of activations as a leader	3998	4181	4158	4082	3786	4159	3851	3793	3901	3765
# of activations as an assistant	81	76	70	64	52	69	41	42	33	31
# of activating actions	4079	4257	4228	4146	3838	4228	3892	3835	3934	3796
# of blocking actions	2490	2637	2612	2537	2428	2545	2201	2241	2182	2071
Total system Actions	6569	6894	6840	6683	6266	6773	6093	6076	6116	5867
Activation rate as Leader	60,86	60,65	60,79	61,08	60,42	61,41	63,20	62,43	63,78	64,17
Activation rate as Assistant	1,23	1,10	1,02	0,96	0,83	1,02	0,67	0,69	0,54	0,53
Blocking rate	37,91	38,25	38,19	37,96	38,75	37,58	36,12	36,88	35,68	35,30

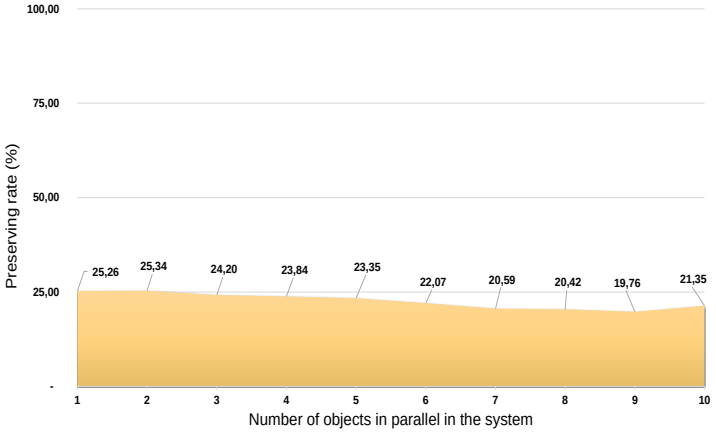


FIGURE 4.30: The conservation rate obtained during the tracking of 500 objects

Figure 4.32 shows that our system has good management of ID switches compared with conventional systems. Indeed, the communication between fog nodes reduces recognition errors when the object leaves the fog surveillance area. In a conventional system, the camera that detects a new arriving object has no idea about the latter's history. However, the proposed system sends all information of the leaving object to the neighbouring fog nodes.

Figure 4.33 shows that the number of tracked objects at more than 80% of their trajectory decreases when the number of objects in parallel increases. This drop in performance was caused by the diminution in the number of assistant cameras. Indeed, the lack of free cameras faced with the increase in the number of objects forces the cameras that play the role of assistant to be released to consider the latter's monitoring, which directly affects the monitoring quality.

TABLE 4.7: Evaluation measures

Measure	Better	Description
False Positives	Lower	Number of objects having crossed the surveillance zone but are not tracked by any camera until leaving the zone
False Alarms	Lower	Number of objects tracked by two or more cameras as different objects
Identity Switches	Lower	Number of object identity changes due to ambiguous or noisy observation
Mostly tracked targets	Higher	Number of objects tracked at more than 80% of their trajectory during their passage through the surveillance zone
Mostly lost targets	Lower	Number of objects tracked at less than 20% of their trajectory during their passage through the surveillance zone

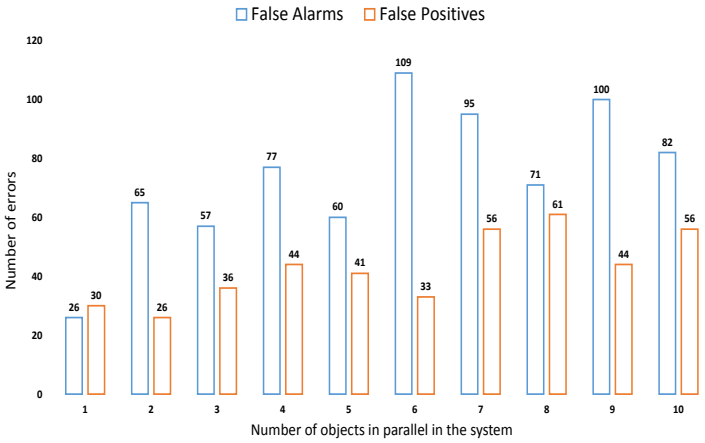


FIGURE 4.31: The average number of errors

4.4.5 Summary and perspectives

This section introduced a novel distributed and collaborative tracking system, which manages the communication of IoVT paradigms based cameras in wireless visual sensor networks. The proposed system aims to increase network life and improve tracking quality by reducing processing costs, energy consumption, and multimedia data traffic. The main idea is to elect one camera as a leader among a group of cameras based on overlapping fields of view to ensure the tracking. The system can also consider the use of assistants to enhance efficiency when the leader's vision is not sufficient to follow the object. The other non-selected neighbouring cameras remain inactive to reduce energy consumption. The proposed solution has achieved good results, increasing the conservation rate by up to 25% in terms of energy, bandwidth, data transmission and storage, which positively impacts the lifetime of wireless visual sensor networks.

During the experimental analysis of our approach, another issue was encountered. This problem resides in the time necessary to find the identifier of the objects in question at the fog nodes, and more precisely at the level of Algorithm 4.15. where the fog node will launch a sequential search on its dataset to find the object identifier (the most similar) in order to re-send it to the requesting cameras and identify the object by the same identifier according to question (RQ1).

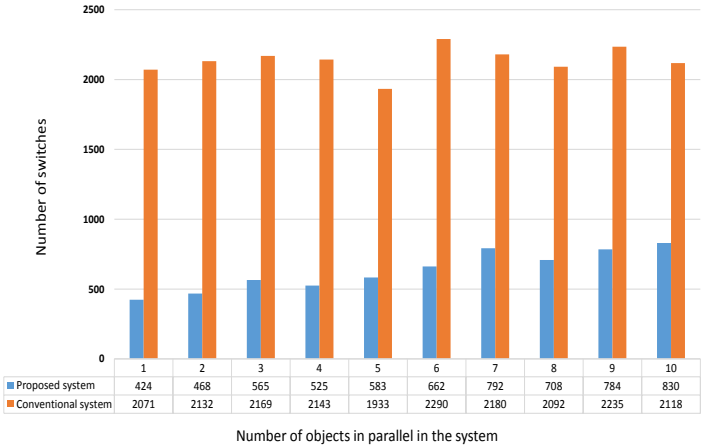


FIGURE 4.32: The total number of switches (ID Sw)

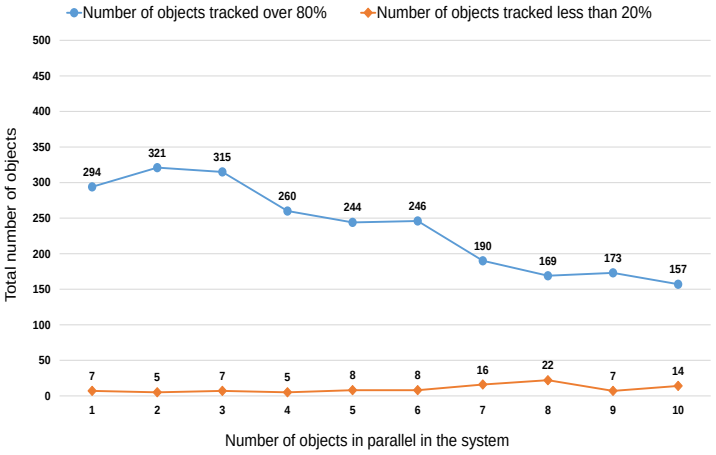


FIGURE 4.33: The ratio of ground-truth trajectories that are covered by a track hypothesis for at least 80% and for at most 20% of their respective life span

Sequential searching basically means that the fog searches through the entire dataset. Therefore, the search time increases relatively with the number of feature vectors loaded into the dataset (linear complexity $O(n)$). With the passage of system operation time, the number of feature vectors increases dramatically depending on the time and the number of objects tracked in that time. Due to this increase, the search time increases; consequently, the waiting time of the requesting camera also increases, where there is a particular occasion when the camera receives the response from fog after the objects are out of their field of vision.

To avoid this issue, we need a new mechanism that allows us to properly organise the data and reduce the search time as much as possible. The main objective of the technique that we need is to improve the search time and avoid the search time's influence on the tracking process, which is a real-time process.

4.5 Conclusion

In this chapter, we have proposed an intelligent video surveillance system for tracking moving objects in a large-scale surveillance area as part of smart city systems based on the emerging

paradigms of IoVT. Our proposed system's principal purposes are to improve tracking quality and reduce overhead costs in terms of energy, bandwidth, and storage.

Before designing this system, several research questions are asked in section (4.1). On this basis, in section (4.2), we present our proposed multi-camera video surveillance architecture based on the IoVT framework. The proposed architecture comprises five layers (Visual sensor layer, mist computing, fog computing, cloud computing, and Application layer). Each layer has specific tasks carried out on it among the tasks of real-time tracking of moving objects. According to the task's time-sensitivity, these tasks are assigned according to the task requirements and the layer nodes' resources.

Our system's coordination mechanism is not applied on all cameras in the system, instead only on sets of cameras that we call clusters or communities. This strategy aims to facilitate the implementation and reduce the coordination mechanism's complexity and restrict the communication area between cameras only at the cluster level. For this purpose, we propose in section (4.3) two methods of grouping cameras. Both methods are based on the cameras' FoV overlapping criterion according to question (RQ4) rather than the distance criterion.

The proposed collaborative approach is based on intelligent cameras that can communicate with other cameras or other equipment and process the generated data. This type of camera allowed us to create an efficient multi-camera coordination mechanism in real-time. This mechanism can manage camera tasks, communication, and information intelligently and efficiently, reduce the number of cameras pandering the tracking without influencing the tracking quality. The proposed system, as shown in section (4.4), operates in two phases: (*1*) *Electing a leader* (or cluster head -CH-) among a set of cameras grouped according to the overlap degree of their FoVs (see question RQ5). (*2*) *Choosing the best assistants* from neighbouring cameras to maximise detection when the leader's vision is not sufficient to track the object. Only the leader and their assistants are active. The other neighbouring cameras remain in an inactive state. In other words, only cameras suitable for tracking objects are allowed to work (question RQ2). The obtained results show that we have reduced the system's operating time by 25% of the total operating time. The latter means that there is no capture, recording, or transmission of data by the cameras at this time, which means less resource consumption (processing, storage, and bandwidth). Therefore, the power consumed and the cost of the system are reduced. Also, the system has significantly reduced the number of ID-switches, which means the tracking process is improved (according to question RQ1).

CHAPTER 5

AN EFFICIENT INDEXING FOR MASSIVE VIDEO SURVEILLANCE DATA BASED ON INTERNET OF VIDEO THINGS -IOVT- PARADIGMS

Chapter contents

5.1 Introduction

At the end of the previous chapter, we mentioned that the sequential search for object re-identification or labelling during tracking gives us a poor performance that negatively affects our proposed system, especially when the data becomes large. This has become a critical problem, especially for large scale multi-camera networks, where the number of visual sensor nodes is enormous, and the amount of multimedia data is correspondingly large (Big IoVT data). The standard search approach is to search not in real multimedia data, but in feature vectors extracted from them [248]. In such environments, an exact match makes little sense, and the concepts of proximity (similarity, dissimilarity) are generally much more helpful for searching [498].

Proximity search has become a fundamental computational task in a variety of application areas, including multimedia information retrieval, data mining, pattern recognition, machine learning, computer vision, and statistical data analysis. With our proposed tracking system, it is easy to see how vital search speed is and how difficult it is to achieve the desired response time. In this case, we need a new mechanism that allows us to properly organise the data and reduce the search time as much as possible. To design such a data organisation mechanism, several questions need to answer.

5.1.1 Research questions and hypotheses

RQ1. ***What is the chosen data organization paradigm?***

In our context, efficient indexing and searching in extensive data collections is one of the most promising paradigms for addressing our issues [498]. Indexing is a data organisation step that should allow efficient access to data when performing similarity queries. Indexing techniques aim to build a data structure that organises the database [130], to provide quick access to the objects in a database by reducing the search space, the cost of input-output, and the number of calculations of distances between objects. In other words, the index provides the efficient implementation of associative search [150].

RQ2. ***Multidimensional space or Metric space?***

Images in our system go through the feature extraction process, so every image is represented as a K-dimensional feature vector. The basic assumption is that the two images are similar according to a distance metric if the corresponding feature vectors are similar.

Therefore, we can say that the image retrieval problem may be considered a multidimensional search problem.

Indexes in multidimensional space are more rigid because of their strong dependency by type or, more specifically, by their geometric properties. This dependency makes the implementation of a flexible structure for a reliable system very difficult. According to Vlastislav Dohna [126], a lot of work has been done on multidimensional space by exploiting its geometric properties. However, index and search techniques for multidimensional spaces usually work well only for low-dimensional spaces. Even indexing methods specifically designed for higher dimensions typically cease to be effective when the dimensionality exceeds twenty.

The metric space approach has been found to be very important in building effective indexes for similarity searching. The most important advantage is that many data types can be indexed because the metric space does not make any requirements on their content or the intrinsic structure of the data, but only a function of distance [271, 498]. In other words, the metric space is more flexible and simpler than the multidimensional space, allowing us to create an index structure that can handle any type of data. Thanks to this advantage, our surveillance system remains more reliable if the cameras' data changes due to a change in the feature vector extraction technique or the cameras' heterogeneity in the system. Another advantage is the possibility of adding other types of data transparently, such as adding different types of sensors in the future. For this reason, we resort to general metric spaces.

RQ3. ***What is the most appropriate indexing structure for large IoVT data?***

According to our review of existing indexing techniques presented in chapter 3, tree indexing structures are the most commonly used in metric space. Tree indexing structures are dynamic structures with data changes. These structures do not require periodic reorganizations of the structure that requires more expensive resources. Instead, it ensures continuous indexing. Thanks to the logarithmic complexity of the insertion and search provided by tree structures, the search time is reduced logarithmically depending on the number of indexed objects. We believe that using the tree structure will be the best choice for our index based on its advantages.

RQ4. ***How to improve the quality of the index structure?***

Most structures suffer from index degradation problems in large-scale data. An inherent insufficient space partitioning in tree-based indexes is the main factor causing this problem, resulting in intra-regional overlap or overlap between tree nodes. These overlapping regions grow rapidly with significant growth in data. Likewise, with the increase of intra-regional overlap, the degeneration of the index structure becomes more widespread. To avoid this problem, according to our study, partitioning data based on hyperplanes is preferred rather than hypersphere-based partitioning (see the last paragraph of section 3.4.2.1). For this reason, this partitioning technique will be used in our structure that we are going to propose.

RQ5. ***How to improve the time and quality of search?***

The objective of the research is to find the desired information quickly and inexpensively. Two main factors directly affect the quality of research:

- *Overlapping partition spheres*

  All Knn search algorithms are based on spheres (the query point/object and the query radius). It is true that the suggested hyperplane partitioning eliminates the

overlapping of partitions but does not guarantee that the spherical modelling of partitions does not overlap. High-overlapping spheres of partitions lead to access to most nodes during the search process. This requires more time as well as more cost of distance calculation, even compared to sequential searching. To solve this problem, we need another technique based on hyperplane partitioning to reduce this overlap. As a preliminary hypothesis, we can use a classification technique such as a data partitioning strategy. Typically, classification techniques aim to split data into clusters to group similar objects together as much as possible and separate dissimilar individuals as much as possible. For this reason, we must shed light on these techniques and consider the possibility of using them and taking advantage of their benefits.

- *Over-partitioning*

  Over-partitioning is another factor we must take into account. Over-partitioning is a problem of unnecessary partitioning of data where similar data will be dispersed. This causes an increase in the height of the tree and the search space (the number of paths traversed in the index structure). To overcome this problem, we need a new strategy that allows us to avoid similar data partitioning as much as possible and maintain the partitions' homogeneity to find similar objects faster.

RQ6. ***How to ensure indexing large-scale data?***

With the new computing paradigms that have emerged in the IoVT arena, distributed indexing systems offer a promising solution to solve the search and discovery problem in Big IoVT data. The latter distributes the system load over the system's different layers (from the sensor to the data centre) through the different emerging paradigms. For the latter solution, several issues must be taken into account to create an efficient indexing structure:

- Is it possible to implement the structure at multiple levels?
- How to partition the indexing system load between these levels?
- To reduce network bandwidth, overall cost, and efficiency, how to select the partition and the steps to be performed?

Data indexing has been selected as a paradigm for organising and managing massive IoVT data (RQ1). According to the search questions, we have defined the indexing structure's main requirements to propose for indexing feature vectors of the proposed surveillance system to improve the time and quality of research and increase the real-time performance tracking process. Consequently, Metric space has been selected as a universal abstraction for data due to the particular advantages discussed in (RQ2). The proposed structure must be a tree structure to benefit from its simplicity, dynamism, and low complexity (RQ3). The tree-based structures are usually based on recursive space partitioning. At this level, the proposed structure should take into account the intra-regional problem through hyperplane-based data partitioning (RQ4). As this overlap increases, the search is likely to degenerate into a complete analysis of the dataset, increasing search time and reducing search quality (RQ5). Also, the structure must take into account the cost of CPU resources and I/Os (RQ5). To adapt this structure to our system, it is essential to adapt it to our architecture proposed in section 4.2. This adaptation allows the proposed structure to benefit from the emerging computing paradigms of the IoVT, which represents the most powerful real-time processing capacity provided by fog computing due to its proximity to cameras and the greater storage capacity offered by cloud computing (RQ6).

The remainder of the chapter is organised as follows. Section 5.2 presents the proposed architecture for the proposed IoVT data indexing mechanism. Section 5.3 explains in detail the proposed index structure. The proposed structure is validated on different types of real data, and the experimental results are presented in section 5.4. Finally, section 5.5 concludes our work and offers some perspectives.

5.2 System architecture overview

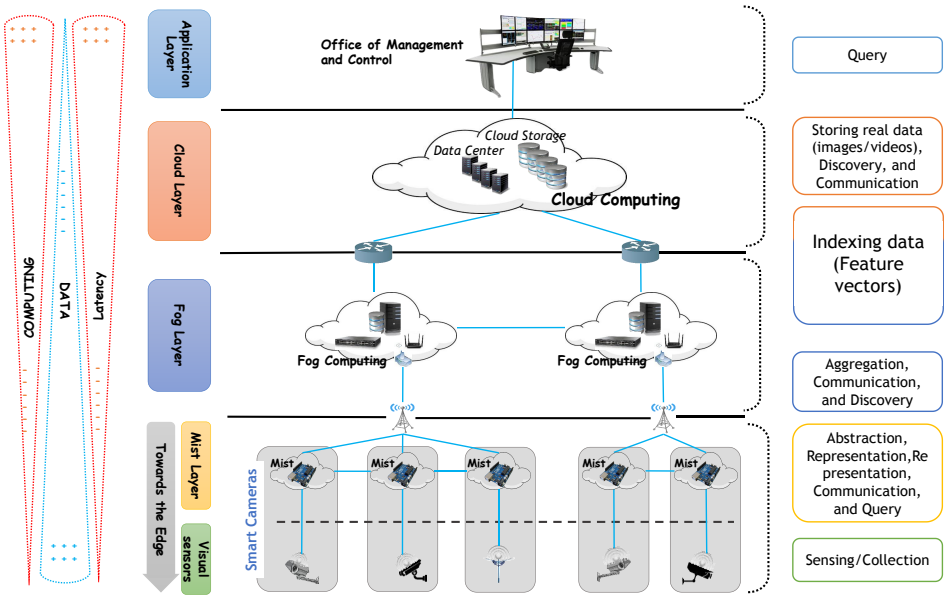


FIGURE 5.1: IoVT data indexing and discovery in cloud-fog-based environment

The goal of the new index structure that we will develop is to accelerate the search for the identification process present in the tracking system proposed in the IoVT environment through an efficient indexing mechanism based on the IoVT paradigms. This mechanism adapts and leverages modern computing paradigms to address data storage, processing and indexing challenges and to improve the quality of research and system/user queries on large-scale IoVT data. Figure 5.1 shows a global view of the cloud-fog-mist computing architecture for the proposed IoVT data indexing mechanism and shows the main elements included in the proposed system with the roles of each in the process of indexing and discovering IoT data which are discussed in [141].

The scenario of the proposed system starts at the visual sensor layer or camera layer. This layer's role is limited to capturing and collecting images/videos of moving objects in the surveillance environment. The next layer is the mist layer. This layer has three main roles in our indexing and discovery process chain: (a) abstraction, (b) representation, (c) communication, and (d) query. Each mist node at this layer processes the collected images and extracts for each detected object its feature vector (according to the proposed tracking system, see Figure 4.2). In our context, feature vector extraction represents a higher-level description of the raw data sufficient to infer information and insights, which are represented axiomatically in machine-readable formats. In our system, transmitting abstracted data instead of all raw data between system elements reduces communication and power consumption in in-network processing.

According to specific protocols, all these mist nodes can transmit and communicate real-time information to carry out intelligent re-organisations, controls, and online monitoring [372]. The mist nodes transmit their generated data to the corresponding fog node. According to our proposed system's tracking process, the mist node can send queries to the fog node at any time to request the ID of a detected object to identify it during tracking.

The role of the fog layer in the proposed IoVT data indexing and discovery process chain is : (a) aggregation, (b) indexing, and (c) discovery (d) communication. Each fog node receives data from all nodes in the lower layer (mist layer) and aggregates it. Then, the fog nodes will index this data in the proposed indexing structure. Our indexing structure is created in this layer to develop an efficient real-time distributed indexing mechanism that allows us to search in real-time. Thus, each fog node has its own structure known as a local structure. The real or raw data (images/videos) indexing in the fog layer is transmitted to the cloud layer to store it.

According to our description of the proposed tracking system in chapter 4, the mist node sends requests to the corresponding fog node to identify the requested objects during the tracking process. To answer these requests, the fog node has the possibility of discovering and searching for this object in its local structure. Suppose the query responses are not available in the local index structure. In that case, the latter re-transmits the received search request to the neighbouring fogs nodes through communication and collaboration between the fogs nodes, as shown in Figure 5.1. Therefore, each neighbouring fog node received the search request, started searching in its local index structure, and re-send the answers to the sending fog node. The latter, in return, aggregates the responses received and chooses the best answer, and forwards them to the requesting user.

In our indexing and discovery process chain, cloud plays the role of (a) indexing, and (b) discovery, in addition to (b) storage and (c) analysis. The real data indexed in the fog layer is stored in the cloud thanks to its large storage capacity and can be used for long-term analysis.

Over time, the amount of data indexing at the fog node reaches its maximum storage capacity. In this case, the fog node transmits its local index structure to the cloud, and in turn, the cloud links the index structure with the actual data that is already stored in it. With the collection of several local indexes of several fog nodes, a global index has been created, which contains all the data of systems in the form of a forest.

At this point, we can identify two scenarios according to the layer where the request was sent. There are cases where the user (Application layer) sends the request directly to the fog layer to obtain simulated objects from a given object only in the corresponding fog region. In this case, the fog node starts the similarity search process on the local index structure. When the fog nodes locate the local index structure's response, it sends requests to the cloud to retrieve the real data searched and re-send it to the user requesting it. The second scenario is when the user sends his request to the cloud, and in this case, the user must retrieve all similar objects of a given object in the overall surveillance area. In this case, the cloud sends the request to each fog node in the surveillance system. Then, each fog node finds similar objects via the local index structure and forwards the cloud response. From these responses, in addition to their result obtained from the search in the global index, the cloud retrieves the real data for each answer and relays it to the user.

5.3 The proposed indexing approach

The problem of exponential data growth in our video surveillance system has led to a requirement for new solutions that allow us to manage and process huge amounts of camera data efficiently and flexibly. Therefore, this section presents a new efficient method for indexing massive data in the multi-camera network, adaptable to our proposed architecture based on IoVT paradigms to improve the quality of the data discovery and retrieval process in real-time.

5.3.1 BCCF-tree definition

The proposed approach called *B*inary tree based on *C*ontainers at *C*loud-*F*og computing level (BCCF-tree). BCCF-tree is a metric space indexing technique that recursively divides the data set into two disjoint regions by selecting two pivots at each iteration. It is a structure similar to the GH-tree in that both partition the data set recursively via the generalised hyperplane principle. The difference is that the BCCF-tree uses different data partitioning techniques, different pivot selection strategies, and different node types.

Firstly, a leaf node E_C consists merely of a subset of the indexed objects stored in a container or bucket. This type of node is deployed to avoid unsatisfactory node division and extend it completely to metric space (RQ5).

$$E_C \subseteq E$$

where, $|E_C| \leq c_{\max}$. is the contents of the leaves of the tree partition E.

Secondly, an inner node N is a sextuple:

$$(p_1, p_2, r_1, r_2, N_L, N_R) \in \mathcal{O} \times \mathcal{O} \times \mathbb{R}^+ \times \mathbb{R}^+ \times \mathcal{N} \times \mathcal{N}$$

where:

- (p_1, p_2) are two *distinct* objects, with $d(p_1, p_2) > 0$, called *pivots*. These pivots are initiated by the two centroids returned by the k-means clustering algorithm during partitioning.
- (r_1, r_2) are the distances to the farthest object in the sub-tree rooted at that node N with respect to p_1 and p_2, respectively, that is, $r_i = \max\{d(p_i, o), \forall o \in N\}$ for $i = 1, 2$, where it is used to help define two balls, $B_1(p_1, r_1)$ and $B_2(p_2, r_2)$, centred on p_1 and p_2, respectively (see the right side of Figure 5.3).
- (N_L, N_R) are two sub-trees (see the left side of Figure 5.3), such that:
  - $N_L = \{o \in N : d(p_1, o) < d(p_2, o)\}$ for the partial ball centred on p_1;
  - $N_R = \{o \in N : d(p_2, o) < d(p_1, o)\}$ for the partial ball centred on p_2;

According to the description of our system architecture in section 5.2, Figure 5.2 shows a distribution of our proposed indexing structure in the cloud-fog computing layer. Each fog node includes a local BCCF-tree created from data transmitted by cameras located in their geographical area.

As shown in Figure 5.2, BCCF-tree structure is composed of two levels, namely:

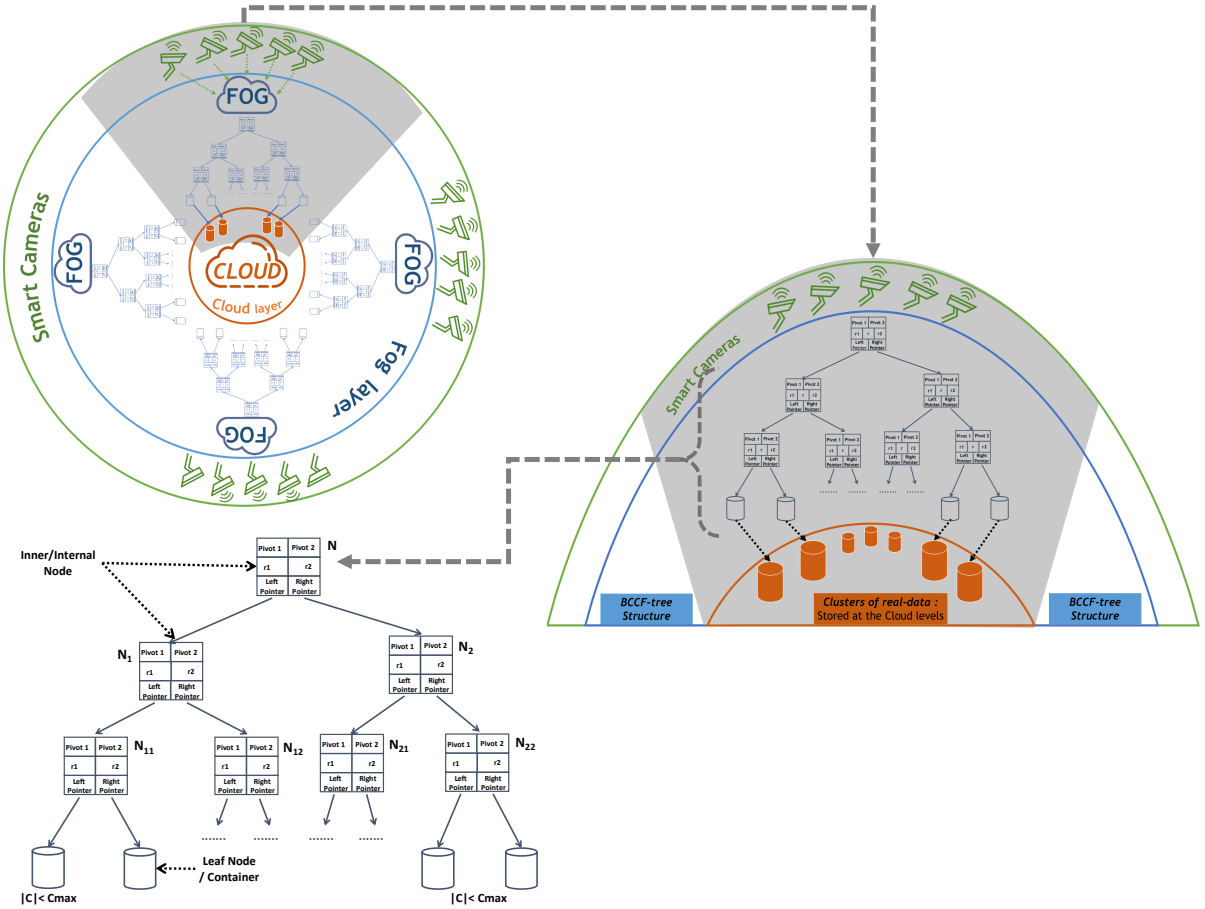


FIGURE 5.2: Cloud-fog-based distribution of the BCCF-tree

- *Internal Node Level* : These nodes contain two pivots (p_1, p_2) and two pointers to the left and right sub-trees. All internal level nodes have empty containers ($|E_C| = \varnothing$), which means that data is not stored in this layer, only the pivots.
- *Leaf Node Level* : These nodes do not contain pivots or pointers but have containers that store a set of objects whose size is less than or equal to $c_{\max}$ ($|N_{C_i}| \leq c_{\max}$).

Figure 5.2 also illustrates that for each leaf node that contains a set of feature vectors at the fog node, there corresponds a cluster containing the real data (images) of the indexed feature vectors in the cloud.

5.3.2 Data partitioning technique

To reduce the overlap of BCCF-tree nodes, BCCF-tree is based on space partitioning through the k-means algorithm (Algorithm 5.1), which allows to group the most similar objects and separate dissimilar objects. Figure 5.3 illustrates the partitioning of data into two non-overlapping regions using the k-means algorithm.

Pivots are objects that are used to eliminate as many objects as possible from the search path. Therefore, selecting pivots represents an essential step. In the literature, many algorithms are based on a random selection (e.g., the first two objects inserted). This arbitrary selection can

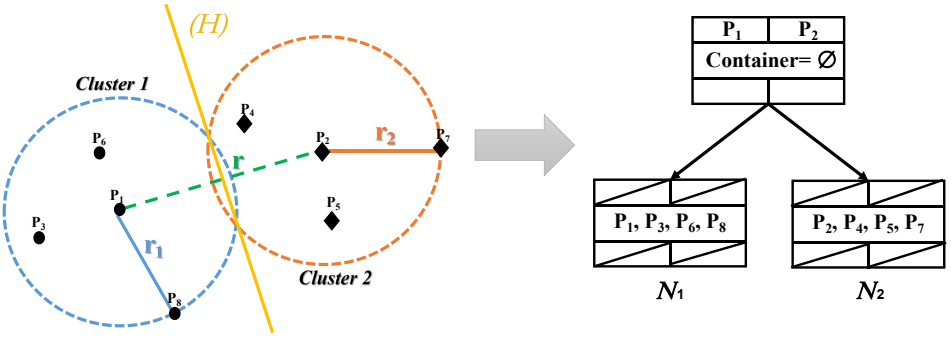


FIGURE 5.3: Partitioning of space with BCCF-tree

affect the performance of search algorithms [219]. As shown in Figure 5.3, at each step of the recursive partitioning process, two pivots are initiated by the centroids retained by the k-means algorithm, where the latter takes the two most distant points from a subset of elements $c_{\max}$ as initial centroids.

The cost of this strategy is reasonable because the search space in the index that we propose does not exceed the square root of the total size of the container ($c_{\max} = \sqrt{n}$). This means that the determination of the two elements furthest from each other in the containers remains linear, which is acceptable. In addition, it is applied only if the container reaches its maximum capacity (RQ5).

5.3.3 Construction of BCCF-tree

The construction of a BCCF-tree is incremental, based on partitioning of data using k-means clustering. The BCCF-tree is a hierarchical directory where the insertion is done from top to bottom. The formal description of the BCCF-tree construction process is represented in algorithm 5.2.

For the insertion of a new object o in the BCCF-tree structure, we travel through the tree until finding the appropriate leaf node to insert it according to the distance $d(o_i, p_i)$ between the inserted object and the pivots of the internal nodes. When the cardinal limit of the container is reached ($|E_C| \geq c_{\max}$), the k-means algorithm (see Algorithm 5.1) is applied over all container objects to partition them into two distinct groups. In the initial step of the k-means process, the two most distant objects that correspond to the initial centroid of the two clusters are chosen. Then, the algorithm places the objects in the nearest centre cluster and recalculates the new centre of each cluster. Finally, the operation is repeated until the algorithm converges. At the end of this process, two well-separated clusters (C_1, C_2) and two centroids (c_1, c_2) are obtained, which have become the pivots (p_1, p_2) of the divided node. Two new leaf nodes are created for both clusters, and each cluster is inserted into a container of one of the corresponding new leaf nodes (see Figure 5.3).

▷ Complexity of BCCF-tree construction

The BCCF-tree construction algorithm's cost on a database of size n can be estimated by two terms. The first term corresponds to the complexity of the k-means clustering algorithm which is estimated based on the size of the partitioned data $c_{\max}$ with $c_{\max} = \sqrt{n}$, the number

Algorithme 5.1 k-means algorithm for BCCF-tree partitioning

Input:
$E = \{e_1, e_2, \cdots, e_{c_{\max}}\}$ ▷ set of $c_{\max}$ objects
$k = 2$ ▷ number of clusters
$c = \{c_1, c_2\}$ ▷ initial centroids: two most distant objects
$Maxiters = 100$ ▷ limit of iterations
Output:
$c = \{c_1, c_2\}$ ▷ the two centroids obtained
$C = \{C_1, C_2\}$ ▷ the two clusters obtained
Begin
    $C = \emptyset$
    $change := false$
    $iter = 0$
    repeat
        $iter+ = 1$
        Assignment Step: assing each object e_i to the cluster with the nearest center

$$\triangleq \begin{cases} C_1^{(t)} \leftarrow e_i & \text{if } \{d(e_i, c_1) \leq d(e_i, c_2) \\ C_2^{(t)} \leftarrow e_i & \text{if } \{d(e_i, c_1) > d(e_i, c_2) \end{cases}$$

        Apdate Step: update the centres
        $c_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{e_j \in C_i} x_j$
        Update the centres with the modified clusters:
        $C^{(t)} = \{C_1^{(t)}, C_2^{(t)}\}$
    until $iter \geq Maxiters \wedge C^{(t)} = C^{(t-1)}$
    return $(c_1^{(t)}, c_2^{(t)}, C_1^{(t)}, C_2^{(t)})$
End.

of clusters m with $m = 2$ (because the BCCF-tree is a binary tree) and the number of the iterations t to converge for a database [494]: $O(2 \cdot t \cdot c_{\max})$. The second term corresponds to the complexity corresponds to the sequence of calculations carried out while going down in the tree during index creation, which gives by $O(n \cdot \log_2(n))$. With the combination of these two terms, we obtained the global complexity of our construction algorithm, which gives the following expression: $O((n \cdot \log_2(n)).2(t \cdot \sqrt{n}))$.

5.3.4 kNN in BCCF-index

Our kNN search algorithm in BCCF-tree is done in two steps:

▷ STEP ONE : The estimation of the radius for the query

To increase search quality and reduce search time in the BCCF-tree, we propose an estimation algorithm that minimises the query radius to reduce the number of leaf nodes visited during the search. Algorithm 5.3 presents the formal description of our algorithm.

The radius is estimated based on the intersection between the query ball $B_q(q, r_q)$ and the internal nodes balls $B_i(p_i, r_i)$ only. At the first, the radius of the query r_q initialized to infinity ($r_q = +\infty$). Next, the estimation process starts to descend to the bottom of the tree as long as

Algorithme 5.2 Construction of BCCF-tree

$$\text{Insertion-BCCF} \left(\begin{array}{l} o \in \mathcal{O}, \\ N \in \mathcal{N}, \\ d : \mathcal{O} \times \mathcal{O} \to \mathbb{R}^+ \end{array} \right) \in \mathcal{N}$$

With:

- $r_L = max\{r_1, d(p_1, o/e)\}$
- $r_R = max\{r_2, d(p_2, o/e)\}$

$$\triangleq \begin{cases} E_C = \{o\} & \text{if } N = \perp \\ Add(o, E_C) & \text{if } N = E_C \wedge |E_C| < C_{max} \\ \left(\begin{array}{l} (e_1, e_2) = \text{Farthest\_objects}(E_C), \\ (c_1, c_2, C_1, C_2) = \text{k-means}(e_1, e_2, E_C), \text{ (algorithm (5.1))} \\ \forall e \in C_1 : (c_1, c_2, r_L, r_2, \text{Insertion-BCCF}(e, L), R) \\ \forall e \in C_2 : (c_1, c_2, r_1, r_R, L, \text{Insertion-BCCF}(e, R)) \end{array} \right) & \text{if } N = E_C \\ (p_1, p_2, \perp, \text{Insertion-BCCF}(e, L), R) & \text{if } N = (p_1, p_2, \perp, L, R) \wedge \\ & d(p_1, o) \leq d(p_2, o) \\ (p_1, p_2, \perp, L, \text{Insertion-BCCF}(e, R)) & \text{if } N = (p_1, p_2, \perp, L, R) \wedge \\ & d(p_1, o) > d(p_2, o) \end{cases}$$

there is an intersection between the query ball $B_q(q, r_q)$ and the sub-tree balls $B_i(p_i, r_i)$. During the descent, the radius is decreased each time it passes over an internal node. This process stops before reaching the leaf nodes or there is no more intersection between the query ball and the sub-tree balls.

▷ STEP TWO : kNN search

Once the estimation algorithm is complete, the kNN search algorithm starts. Algorithm 5.4, formally describes the search kNN in a BCCF-tree. The searches are made from balls while data has been partitioned. The search is done by calculating the intersection between the query ball, which was initialised by the value estimated in the first step, and the two sub-tree balls. While the intersection between the balls is present, the search process keeps going down until it reaches the candidate leaf nodes.

The leaf nodes contain a subset of the indexed data with a maximum cardinal $c_{\max}$. At the leaf level, the algorithm is quite simple. To find the k closest neighbours of a leaf, they have to be sorted according to their increasing distances to the q request object. Then, we return *at most* the first k objects already sorted. Note that an actual sort is not necessary; there is a variant, called "k-tri", which is only in $O(c_{\max} \cdot \log_2(k))$, rather than, $O(c_{\max} \cdot \log_2(c_{\max}))$. The complexity of the operation on a sheet is very fast, since $c_{\max}$ is a logarithm of the collection's size.

▷ Complexity of kNN research in BCCF-tree

Let us consider the case $c_{\max} = \sqrt{n}$, the time complexity is in the order:

Algorithme 5.3 Estimated radius of query r_q for kNN search in BCCF-Index

$$\text{Estimated-}r_q \left(\begin{array}{l} N \in \mathcal{N}, \\ q \in \mathcal{O}, \\ d : \mathcal{O} \times \mathcal{O} \to \mathbb{R}^+, \\ r_q \in \mathbb{R}^+ = +\infty \end{array} \right) \in (\mathbb{R}^+ \times \mathcal{O})^{\mathbb{N}}$$

With:

- $r_q^L = min\{r_q, (d(q, p_1) + r_1)\}$
- $r_q^R = min\{r_q, (d(q, p_2) + r_2)\}$
- $r_q^{LR} = min\{r_q, max\{(d(q, p_1) + r_1), (d(q, p_2) + r_2)\}\}$
- $r_q^{E_L} = \text{Estimated-}r_q(N_L, q, d, r_q^L)$ ▷ $B_q(q, r_q) \cap B_{N_1}(p1, r_1)$
- $r_q^{E_R} = \text{Estimated-}r_q(N_R, q, d, r_q^R)$ ▷ $B_q(q, r_q) \cap B_{N_2}(p2, r_2)$
- $r_q^{E_{LR}} = min\{\text{Estimated-}r_q(N_L, q, d, r_q^{L_R}), \text{Estimated-}r_q(N_R, q, d, r_q^{L_R})\}$
  ▷ $B_q(q, r_q) \cap B_{N_1}(p1, r_1) \quad \wedge \quad B_q(q, r_q) \cap B_{N_2}(p2, r_2)$

$$\triangleq \begin{cases} r_q & \text{if } N = \perp \\ r_q & \text{if } N = E_C \\ r_q^{E_L} & \text{if } N = (p_1, p_2, r_1, r_2, N_L, N_R) \wedge d(p_1, q) < (r_q + r_1) \wedge d(p_2, q) \geq (r_q + r_2) \\ r_q^{E_R} & \text{if } N = (p_1, p_2, r_1, r_2, N_L, N_R) \wedge d(p_2, q) < (r_q + r_2) \wedge d(p_1, q) \geq (r_q + r_1) \\ r_q^{E_{LR}} & \text{if } N = (p_1, p_2, r_1, r_2, N_L, N_R) \wedge d(p_1, q) < (r_q + r_1) \wedge d(p_2, q) < (r_q + r_2) \end{cases}$$

$$O\left(\left(\tfrac{1}{2}\sqrt{n}.log_2(k)\right) + \left(\left(log_2\left(\tfrac{n}{\frac{1}{2}\sqrt{n}}\right)\right) \cdot 2k\right)\right), \text{ with: } n = |E|.$$

The first term $O(\frac{1}{2}\sqrt{n}.log_2(k))$ corresponds to the calculations carried out on the leaf, where the first factor $\frac{1}{2}\sqrt{n}$ estimates that the leaves are half-filled on average. The second term $((log_2(n)/(\frac{1}{2}\sqrt{n})) \cdot 2k)$ corresponds to the computations sequence carried out while going up in the index, namely by multiplying the tree's height $\left(\frac{n}{\frac{1}{2}\sqrt{n}}\right)$ by fusion time $(2k)$. We consider that k is a "constant", in the sense that its value is independent of the collection size and probably much smaller than $\sqrt{n}$.

5.4 Experiments and results

The objective of the BCCF-tree is to build an index and to respond efficiently to queries. This section presents an experimental evaluation of this structure by comparing it with other indexing methods.

5.4.1 Simulation setup

The prototype was implemented in Python language on a workstation with an Intel® Core^{TM} i5, 4200U CPU, 1,6 GHz processor, 4 GB RAM memory capacity and have Linux (Ubuntu)

Algorithme 5.4 Search for kNN in BCCF-Index

$$k\text{NN-BCCF-index}\left(\begin{array}{l} N \in \mathcal{N}, \\ q \in \mathcal{O}, \\ k \in \mathbb{N}^*, \\ d : \mathcal{O} \times \mathcal{O} \to \mathbb{R}^+, \\ r_q \in \mathbb{R}^+ \\ A \in (\mathbb{R}^+ \times \mathcal{O})^{\mathbb{N}} = \emptyset \end{array}\right) \in (\mathbb{R}^+ \times \mathcal{O})^{\mathbb{N}}$$

With:

- $A^C = k\text{-sort}(A, (d(e,q),e)|e \in E_C)$
- $A^L = k\text{NN-BCCF-index}(L, q, k, r_q, A))$ ▷ $B_q(q, r_q) \cap B_{N_1}(p1, r_1)$
- $A^R = k\text{NN-BCCF-index}(R, q, k, r_q, A))$ ▷ $B_q(q, r_q) \cap B_{N_2}(p2, r_2)$
- $A^{LR} = k\text{marge}(A^L, A^R)$ ▷ $B_q(q, r_q) \cap B_{N_1}(p1, r_1) \wedge B_q(q, r_q) \cap B_{N_2}(p2, r_2)$

$$\triangleq \begin{cases} A & \text{if } N = \perp \\ A^C & \text{if } N = E_C \\ A^L & \text{if } N = (p_1, p_2, r_1, r_2, N_L, N_R) \wedge d(p_1, q) < (r_q + r_1) \wedge d(p_2, q) \geq (r_q + r_2) \\ A^R & \text{if } N = (p_1, p_2, r_1, r_2, N_L, N_R) \wedge d(p_2, q) < (r_q + r_2) \wedge d(p_1, q) \geq (r_q + r_1) \\ A^{LR} & \text{if } N = (p_1, p_2, r_1, r_2, N_L, N_R) \wedge d(p_1, q) < (r_q + r_1) \wedge d(p_2, q) < (r_q + r_2) \end{cases}$$

TABLE 5.1: Database Characteristics

Dataset	***# of Objects***	***Dimensions***	$c_{max} = \sqrt{n}$
BD-L-TC	990	2	31
GPS Trajectory	18107	3	134
Tracking Data	62702	20	250
WARD	3078552	5	1754

as the operating system. Four datasets were used for the experiments, and Table 5.1 presents these datasets' characteristics.

DB1 : The first dataset is GPS Trajectory[1] which contains the trajectories of GO!Track application users using a variety of means of transport in north-east Brazil [115].

DB2 : The second dataset contains a list of locations and other places with their geographical coordinates obtained from the BD-L-TC topographic database[2].

DB3 : The third dataset represents moving object feature vectors obtained by our object tracking simulator developed presented in chapter 4.

DB4 : The last dataset, called Wearable Action Recognition Database (WARD)[3], is a benchmark database for recognising human action using a wearable motion sensor network [477].

[1]GPS: https://archive.ics.uci.edu/ml/datasets/GPS+Trajectories
[2]BD-L-TC: https://data.public.lu/fr/datasets/r/a7d551d7-f374-491a-ab93-63715b98e6dd
[3]WARD: https://people.eecs.berkeley.edu/~yang/software/WAR/WARD1.zip

Several experiments were carried out to evaluate the proposed indexing structure (BCCF-tree), which is based on containers compared to the following three types of trees:

- **Generalised Hyper-plane:** GH-tree is a binary tree [424] based on the principle of metric space partitioning using hyper-planes. This technique's principle is the recursive partitioning of data into two groups according to the two pivots chosen in each partitioning according to the furthest distance.
- **Bubble Buckets tree:** BB-tree is a new multidimensional index structure proposed by Sprenger et al. [395, 396] that combines the kD-tree [148] and X-tree [56] structure. This structure is based on the recursive partitioning of the data space into k partitions, as for kD-tree [53]. BB-tree uses the structure of bubble buckets with limited capacity (bmax) in the leaf nodes of the tree-like the Adaptive kD-tree, which stores the data (sub-set) and in the case of saturation, the latter is transformed into supernodes similar to X-tree where the data from these nodes are scanned linearly.
- **MX-tree:** Murgante et al. [203] proposed a new metric indexing structure called MX-tree based on the original M-tree structure [104, 105]. MX-tree adapts the concept of supernodes inspired by X-tree structure that avoids the unsatisfactory division of nodes, reducing the cost of calculation and extending it completely to metric space where the time complexity is reduced to $O(n^2)$ without tuning any parameter. Unlike M-tree, where time complexity is waiting for $O(n^3)$.

5.4.2 Evaluation of the index structure

This section discusses the quality of the index on different real data; different measures were conducted, such as the number of containers and the distribution of objects, the height of the tree, the number of internal nodes, the number of nodes in each level.

Figures 5.4, 5.5 and 5.6 show the different experiences in evaluating the structure of the BCCF-index compared to GH-tree, BB-tree and MX-tree. Figure 5.4 presents the distribution of objects in containers at the leaf level for each collection. Figure 5.5 summarises the number of nodes at each level of the tree for each data collection. The number of leaf nodes, which represents the number of containers in the tree, the maximum height of the index tree, and the number of internal nodes for each tree created in each collection are shown in Figure 5.6.

The distribution of data and the number of nodes per level in BCCF-tree, which are indicated in Figures 5.4 and 5.5, respectively, illustrate that the BCCF-tree structure is balanced, which means that the proposed index structure can index even more objects. The Figure 5.6 also shows that the structure of the BCCF-tree has a slightly higher height, number of internal nodes and number of leaf nodes than BB-tree and MX-tree. This increase is due to the efficient partitioning of the BCCF-tree data using the k-means clustering algorithm.

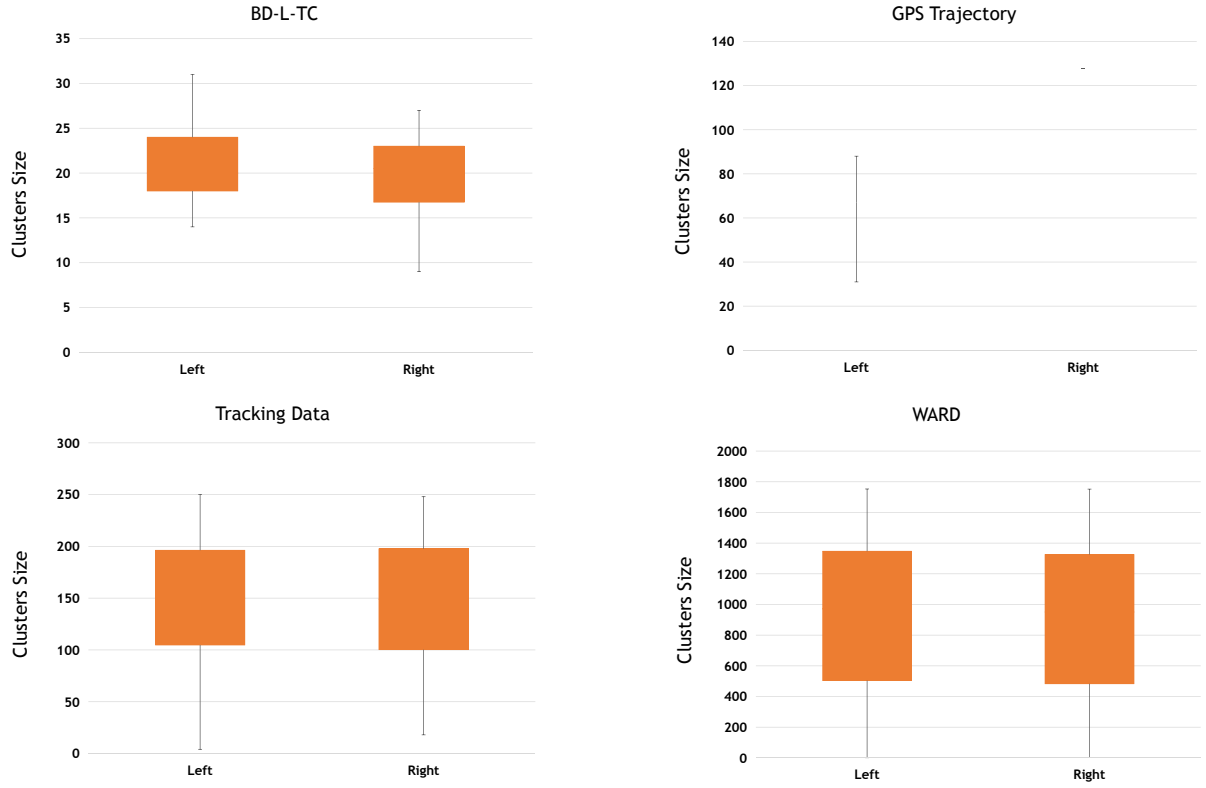


FIGURE 5.4: Distribution of data in the BCCF-tree

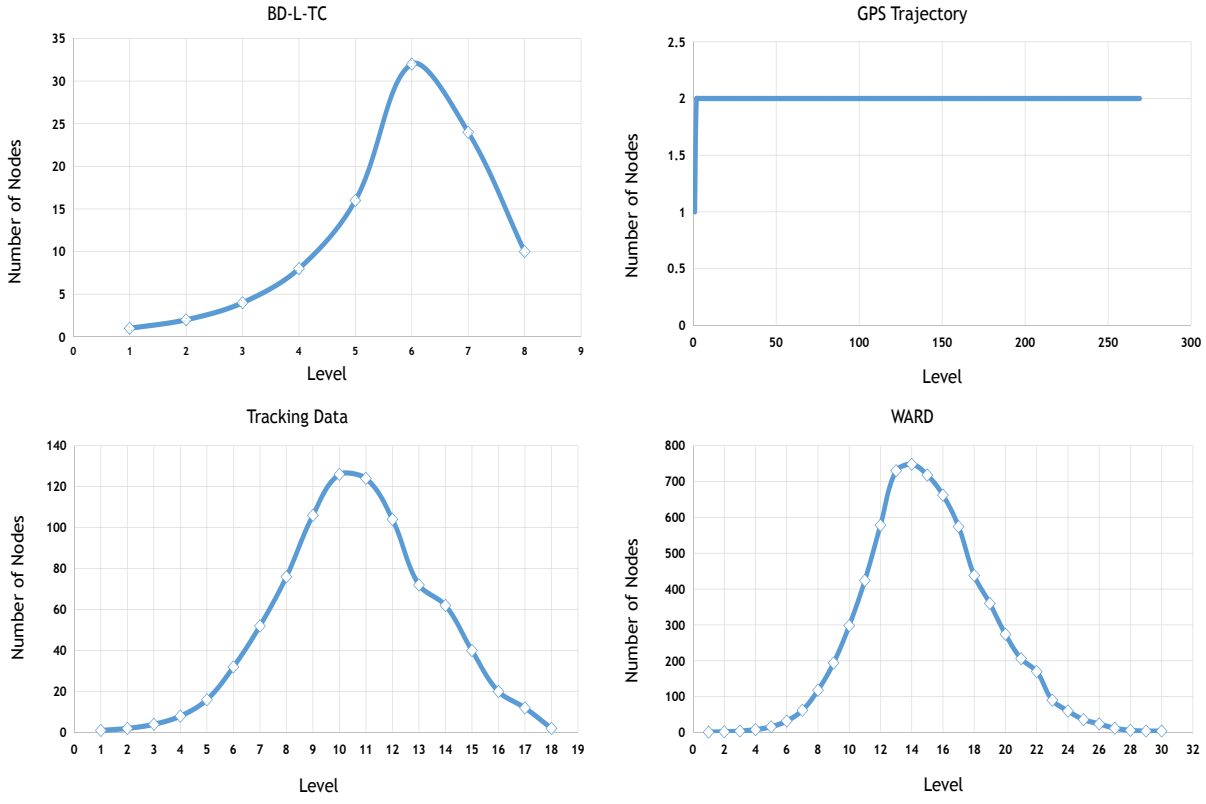
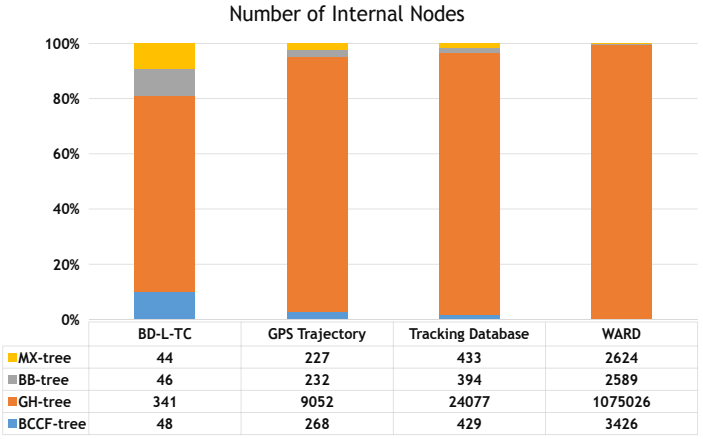
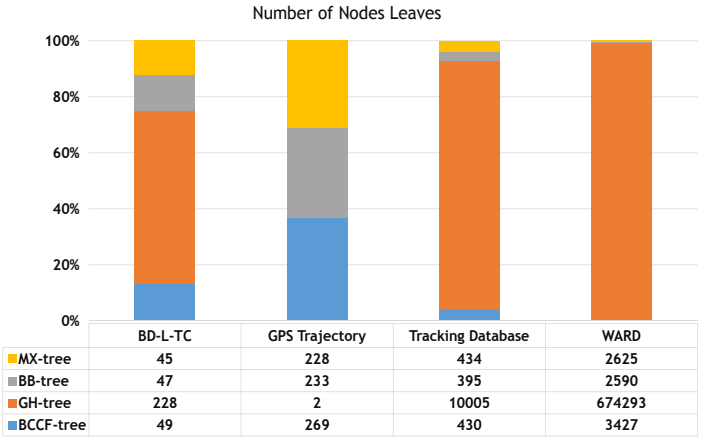


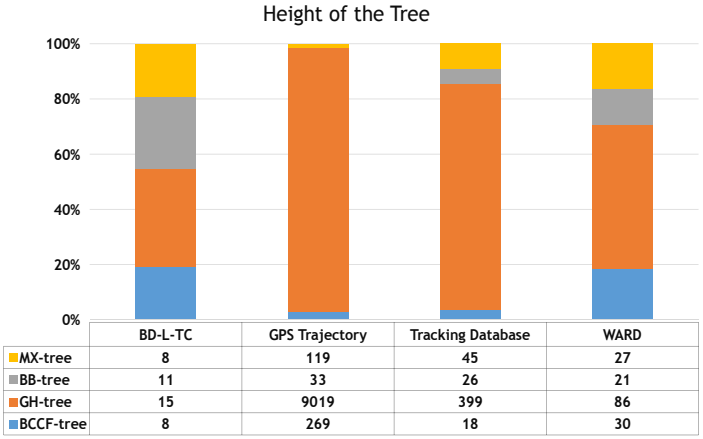
FIGURE 5.5: Number of nodes per level in the BCCF-tree



	BD-L-TC	GPS Trajectory	Tracking Database	WARD
MX-tree	44	227	433	2624
BB-tree	46	232	394	2589
GH-tree	341	9052	24077	1075026
BCCF-tree	48	268	429	3426



	BD-L-TC	GPS Trajectory	Tracking Database	WARD
MX-tree	45	228	434	2625
BB-tree	47	233	395	2590
GH-tree	228	2	10005	674293
BCCF-tree	49	269	430	3427



	BD-L-TC	GPS Trajectory	Tracking Database	WARD
MX-tree	8	119	45	27
BB-tree	11	33	26	21
GH-tree	15	9019	399	86
BCCF-tree	8	269	18	30

FIGURE 5.6: Evaluation of the index structure

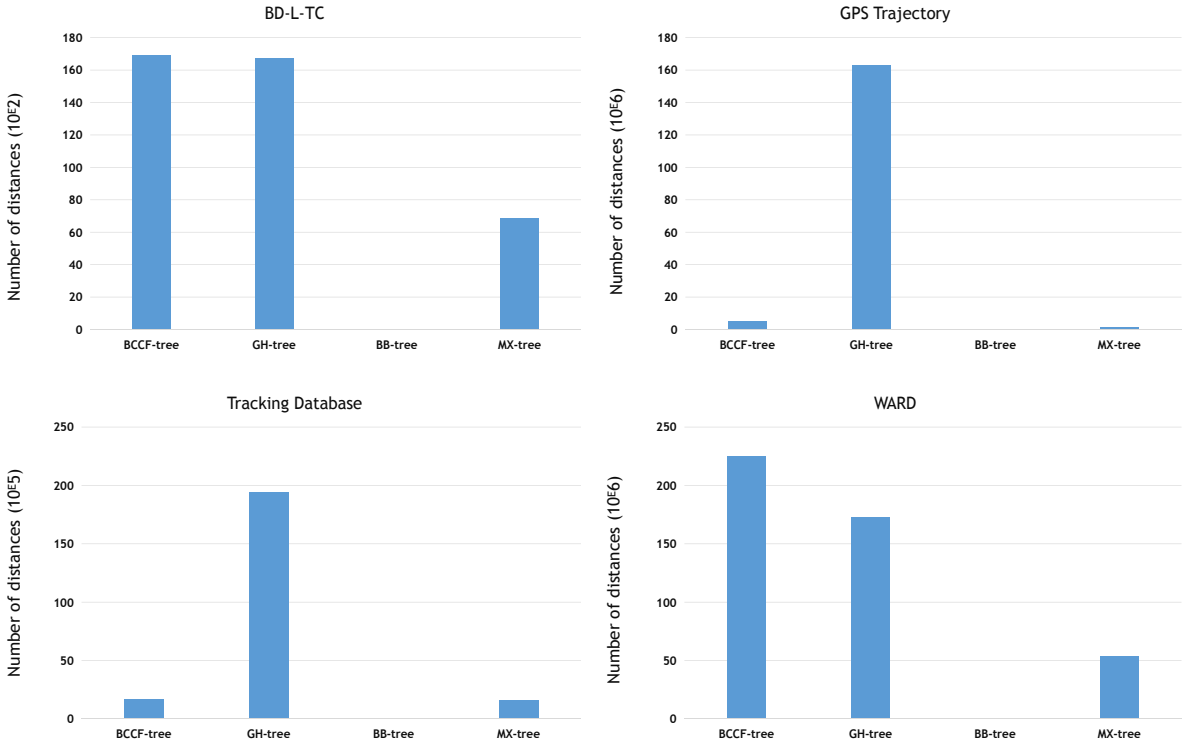


FIGURE 5.7: Number of distances calculated for constructing the BCCF, GH, BB, and MX trees

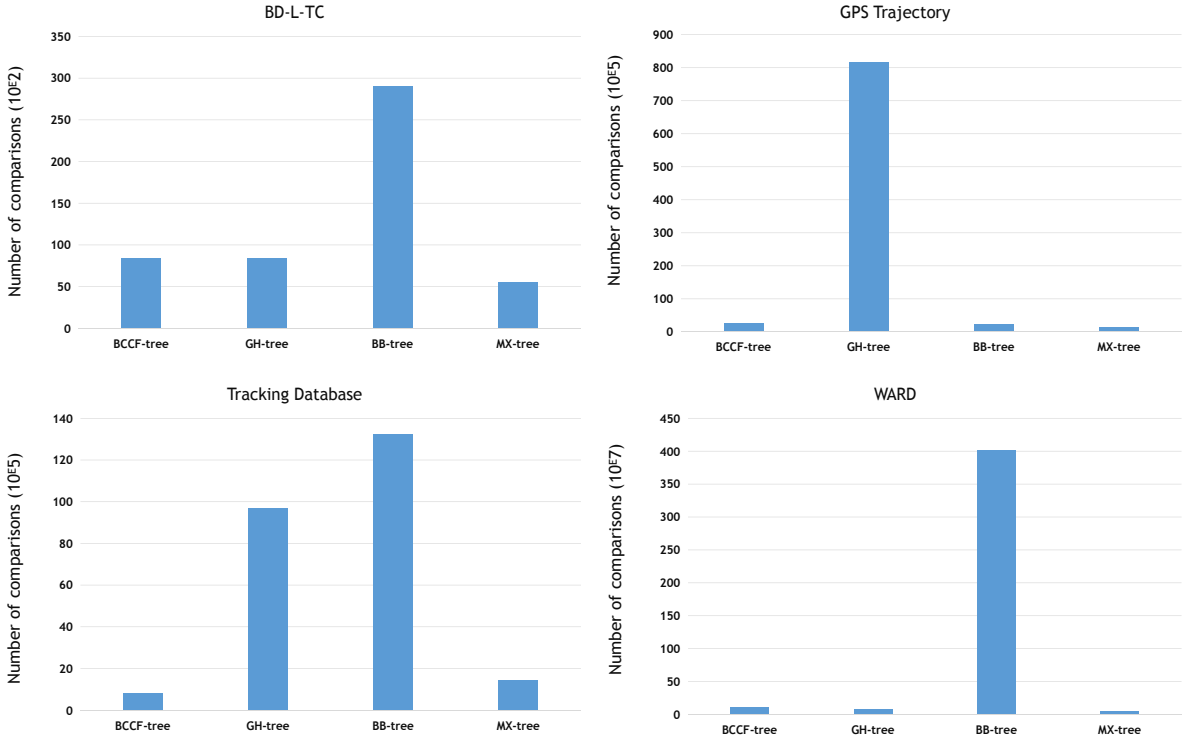


FIGURE 5.8: Number of comparisons performed for constructing the BCCF, GH, BB, and MX trees

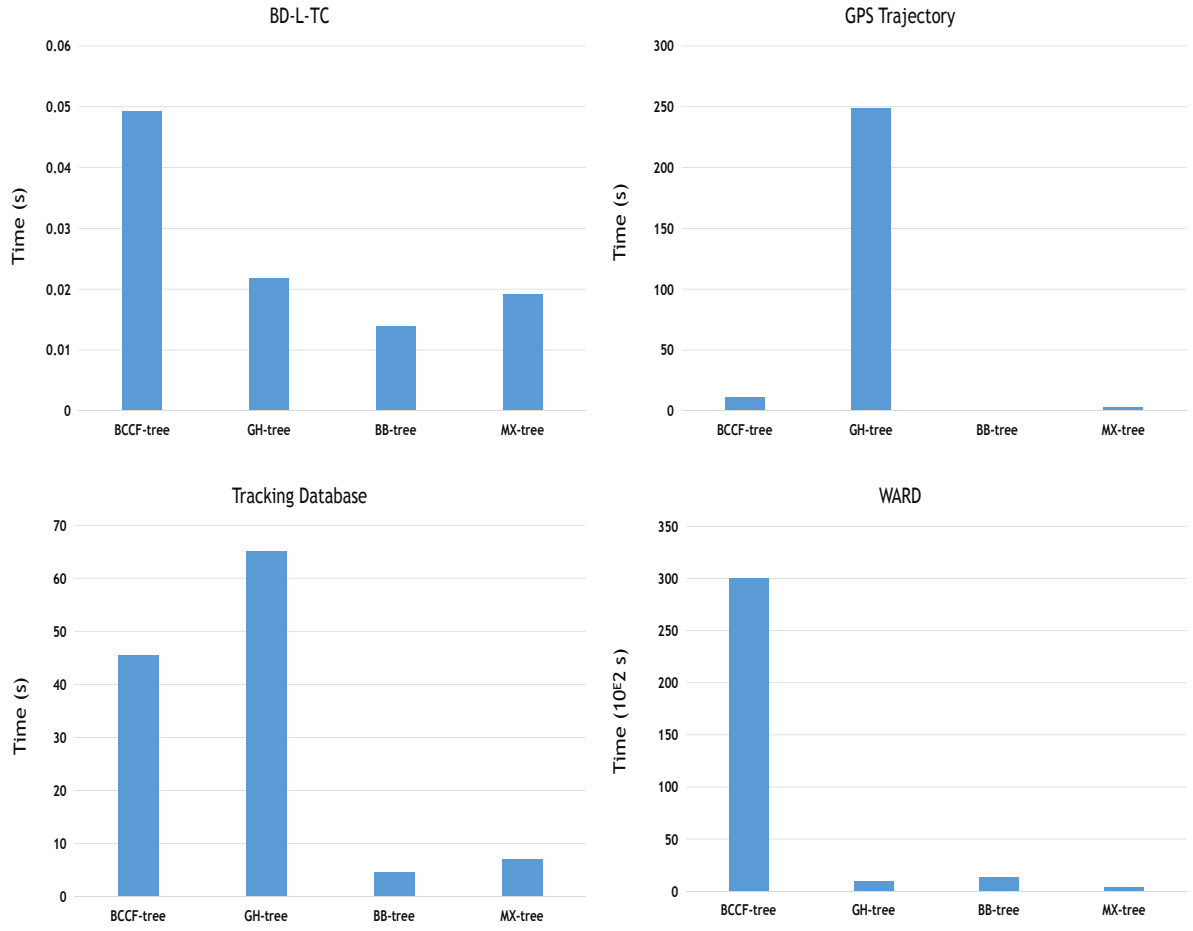


FIGURE 5.9: The construction time of BCCF, GH, BB, MX trees

The number of distances calculated, the number of comparisons made and the time required during the construction of the BCCF-tree are determined to evaluate the creation of the proposed structure compared with the above three structures. The results obtained are presented in Figures 5.7, 5.8, and 5.9, where the size of the container is fixed by $c_{max} = \sqrt{n}$ (n is the size of the database). The cost of the construction of all structures is based on the comparison with the pivots of each internal node until the object is inserted into the corresponding region. In BCCF-tree, the choice of initial centroids for the k-means algorithm where these centroids correspond to the two most distant points and the cost of the convergence of the k-means algorithm when the container is partitioned.

The results show that BCCF-tree is the most efficient than GH-tree in all collections except the WARD benchmark database. Regarding BB-tree and MX-tree, they are the best compared to BCCF-tree and GH-tree. The main factor causing the deterioration of BCCF-tree's performance results is the convergence cost of the k-means algorithm was high due to the number of times partitioning and container sizes.

5.4.3 Evaluation of the exact search

This section presents the proposed structure's performance in terms of exact search, where algorithm 5.5 describes the formal description of the exact search in BCCF-tree. The exact search algorithm is an algorithm that allows finding the best similar point to the requested point in the index tree. This algorithm is similar to the kNN search (algorithm 5.4) but without an estimation step. The complexity of this algorithm is shown in the following expression:

$$O\left(2k \cdot \left(log_2\left(\frac{n}{\frac{1}{2}\sqrt{n}}\right)\right)\right)$$

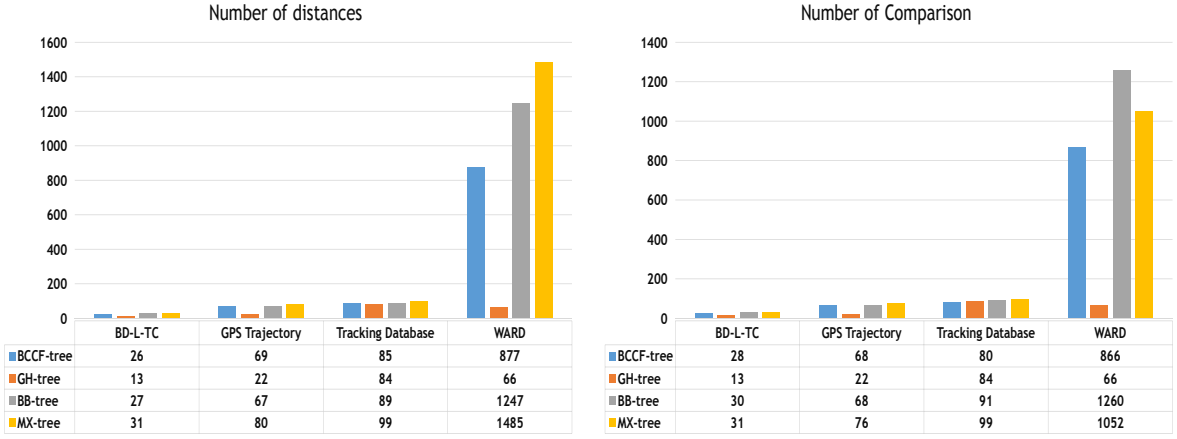


FIGURE 5.10: Number of distances calculated, comparisons performed and the execution time of the exact search in BCCF, GH, BB, MX trees

The results of experiments of the exact search algorithm presented in Figure 5.10 and Table 5.2 show that the GH-tree structure is the most efficient than all the other structures. The high performance of the GH-tree is caused by the absence of containers (in BCCF-tree) or Buckets (in BB-tree and MX-tree), where the exact search process takes only one path in the tree to find the answer. On the other hand, in the other structures, the exact search process also takes a single path plus the linear search of objects in the leaf node containing a set of maximum size data equal

TABLE 5.2: The exact values of the calculated number of distances, comparisons made and the execution time for the exact search in BCCF, GH, BB, MX trees

	Number of distances				**Number of comparison**				**Times (10^{-5}s)**			
	BCCF	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**
DB1	26	13	27	31	28	13	30	31	3,71	1,23	3,50	3,88
DB2	69	22	67	80	68	22	68	76	9,70	3,83	8,44	9,92
DB3	85	84	89	99	80	84	91	99	30,77	30,94	32,75	34,30
DB4	877	66	1247	1485	866	66	1260	1052	150,87	9,68	211,08	158,38

c_{max}. Comparing the other three structures, we find that our BCCF-tree structure achieved more efficiency than BB-tree and MX-tree in terms of distance, comparison and execution time.

Algorithme 5.5 Exact Search in BCCF-Index

$$\text{ExactSearch}\left(\begin{array}{l} N \in \mathcal{N}, \\ q \in \mathcal{O}, \\ d : \mathcal{O} \times \mathcal{O} \to \mathbb{R}^+, \\ A \in \mathcal{O} \end{array}\right) \in (\mathbb{R}^+ \times \mathcal{O})^{\mathbb{N}}$$

With:

- $A^C = \text{1-sort}(A, (d(e,q), e) | e \in E_C)$
- $A^L = \text{ExactSearch}(L, q, A)$
- $A^R = \text{ExactSearch}(R, q, A)$

$$\triangleq \begin{cases} \emptyset & \text{if } N = \perp \\ A^C & \text{if } N = E_C \\ A^L & \text{if } N = (p_1, p_2, r_1, r_2, N_L, N_R) \wedge d(p_1, q) < d(p_2, q) \\ A^R & \text{if } N = (p_1, p_2, r_1, r_2, N_L, N_R) \wedge d(p_2, q) < d(p_1, q) \end{cases}$$

5.4.4 Evaluation of kNN search

The similarity search algorithm's performance in the proposed BCCF-tree structure was evaluated compared to the GH-tree, BB-tree and MX-tree structures. The average number of distances calculated, the number of comparisons performed, and the execution times to meet the 100 requests kNN with different parameter values were determined to interpret better the experimental results k (k=5, k10, k=15, k=20, k=50, k=100). Figure 5.11, Figure 5.12 and Figure 5.13 shows the results of the kNN research experiments. Table 5.3, Table 5.4 and Table 5.5 show the exact values of these results.

The simulation results show that the number of distances calculated, the number of comparisons made, and the search times are also correlated with the increase in the size of the data set and the increase in the k parameter. The results shown in Figure 5.11 show a stabilisation of the number of distances in the GH-tree regardless of the k value. This value is a pocket of the number of elements in the data set, which explains why the search algorithm crosses the entire tree, so the GH-tree does not support the kNN search. Concerning the BB-tree and the MX-tree is strongly influenced by the k value, or when the k value is increased, the number of distances is considerably reduced as shown in Figure 5.11. On the other hand, we see a slight increase in the number of distances calculated for the BCCF-tree, which means that the change in the k value has no significant influence on the search algorithm's performance on our BCCF-tree structure. For the size of the database, we observe the same observation, where the size of the database influences research performance, except the BCCF-tree.

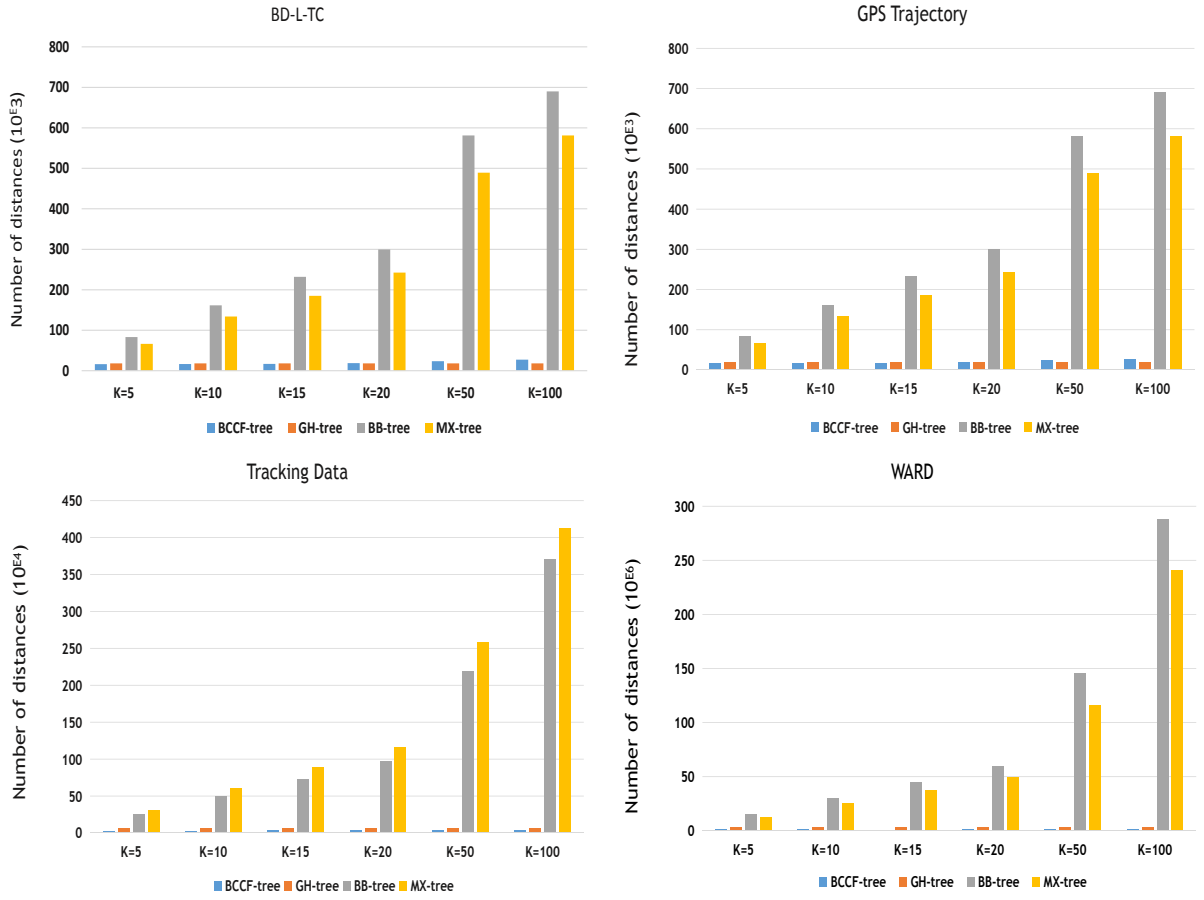


FIGURE 5.11: Number of distances calculated for the kNN search in BCCF, GH, BB, MX trees

Figure 5.12 shows that the GH-tree structure performs a significant number of comparisons during the research process, where the latter influences the similarity research performance. The comparison number performed is relative to the value of k and the size of the database. The same for BCCF-tree, but the increase in the comparison number is much less than GH-tree. Concerning BB-tree and MX-tree, we observe that these two structures are the best performance in comparison number.

To improve the interpretation of experimental results, we calculate the execution time of research processes and the results are shown in Figure 5.13. According to the results presented in this figure, we see that our proposed structure is the most efficient than others in terms of research time. When we exclude the GH-tree structure that gives us poor results, and as we previously stated that this structure does not support the similarity search kNN, we find that BCCF-tree is more efficient than BB-tree and MX-tree respectively by 70%, 61% in BD-L-TC dataset, 42%, 55% in Tracking Data and 70%, 68% in WARD benchmark database. Based on these results, our structure's efficiency rate is higher in the WARD benchmark than other data sets, which allows us to conclude that the proposed BCCF-tree structure is more efficient for large-scale IoT data processing.

TABLE 5.3: The exact values of the distance number computed by the kNN search in the BCCF, GH, BB and MX trees

	BD-L-TC				**GPS Trajectory**				**Tracking Data**				**WARD**			
	BCCF	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**
k=5	392	988	4340	3553	16263	18107	83274	66457	26575	62702	251929	307738	968991	3078552	15143569	12634390
k=10	440	988	7440	5953	16712	18107	161526	133886	26482	62702	491953	603814	1023250	3078552	30488612	25461162
k=15	496	988	9390	8111	16801	18107	232093	184932	28418	62702	725635	888979	234389	3078552	44802908	37301531
k=20	527	988	10260	8712	18691	18107	299469	242396	29069	62702	964322	1163256	1045212	3078552	59234467	49689978
k=50	573	988	10551	9488	23448	18107	581393	489196	32801	62702	2192628	2580617	1077933	3078552	145834746	116428293
k=100	599	988	10551	9488	27354	18107	690120	581253	39743	62702	3708435	4117856	1110101	3078552	288000986	240783208

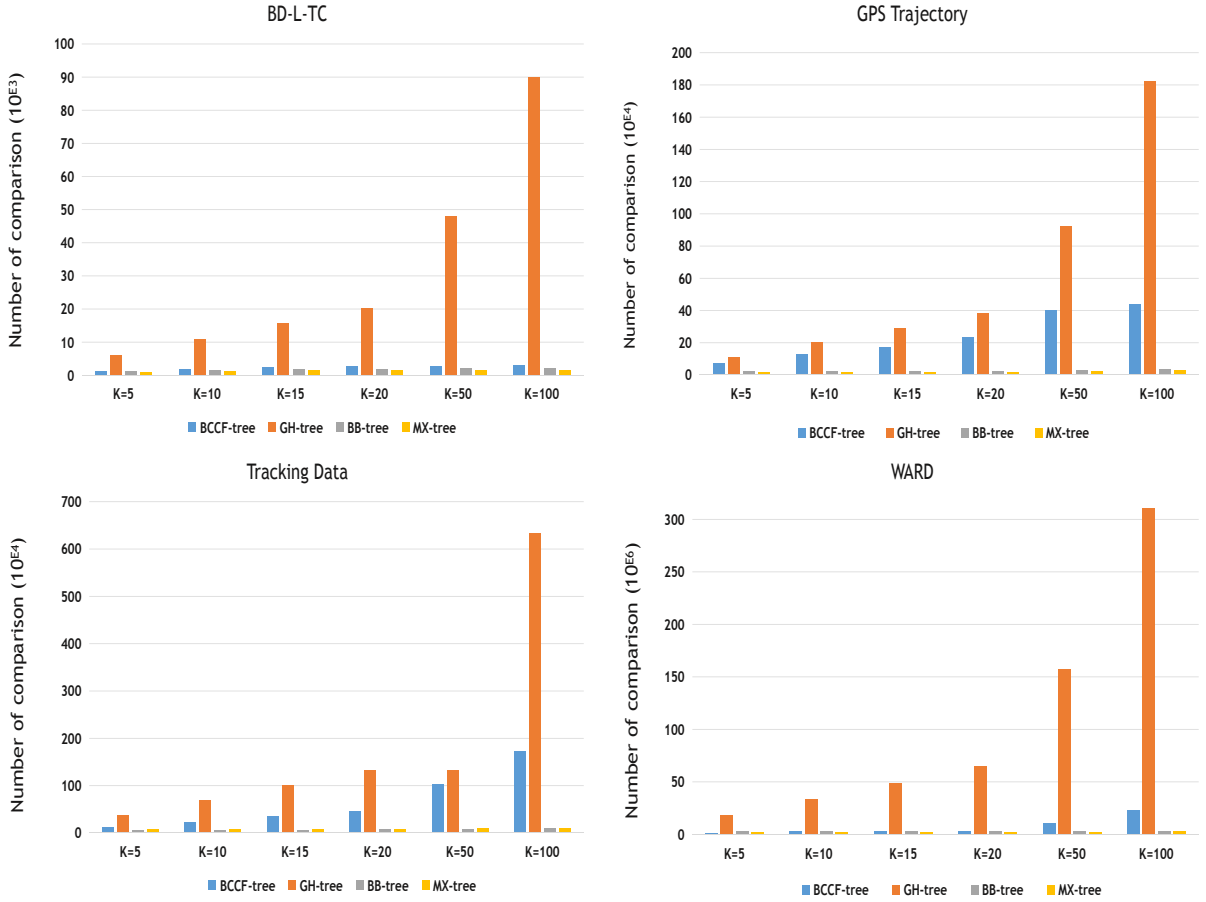


FIGURE 5.12: Number of comparisons performed for the kNN search of BCCF, GH, BB, MX trees

The efficiency of our structure is explained by the number of leaf nodes visited during the research process presented in Figure 5.14. When looking at Figure 5.13 and Figure 5.14, we observe a correspondence relationship between the number of sheets visited. The execution time and the reduction in the number of nodes visited during the search in our structure compared to other structures is due to the efficient partitioning of the data generated by the k-means clustering algorithm, where the latter reduces the overlap between the nodes in the tree.

Regarding the GPS Trajectory dataset results, the MX-tree is more efficient than the BCCF-tree; this is due to the lousy structure created as shown in Figures 5.4 and 5.5, explained by a high convergence between the trajectory points. This influences the interval of the search process kNN where, regardless of the diameter of the estimated query ball, it remains overlapped with most regions of the BCCF-tree (Figure 5.14).

TABLE 5.4: The exact values of the comparison number performed by the kNN search in the BCCF, GH, BB and MX trees

	BD-L-TC				**GPS Trajectory**				**Tracking Data**				**WARD**			
	BCCF	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**
k=5	1104	5908	1308	1043	68571	108622	18802	14892	121843	376192	58158	65656	1061400	18471292	3095122	2547301
k=10	1863	10778	1538	1164	126070	199087	20032	16426	231292	689632	59114	67842	3081307	33863982	3108810	2583356
k=15	2432	15598	1768	1404	171052	289502	21011	16517	355050	1003022	60752	70016	3302929	49256622	3119320	2539323
k=20	2683	20368	1958	1471	230359	379867	22212	17683	464292	1316362	65121	72204	3575209	64649212	3131327	2553167
k=50	2853	47938	2066	1652	400529	921007	28929	23439	1025041	1316362	72413	85195	10721229	157003702	3209920	2484902
k=100	2994	89888	2066	1652	440040	1818907	34327	27035	1718080	6323002	85439	104660	23091773	310923852	3341227	2743101

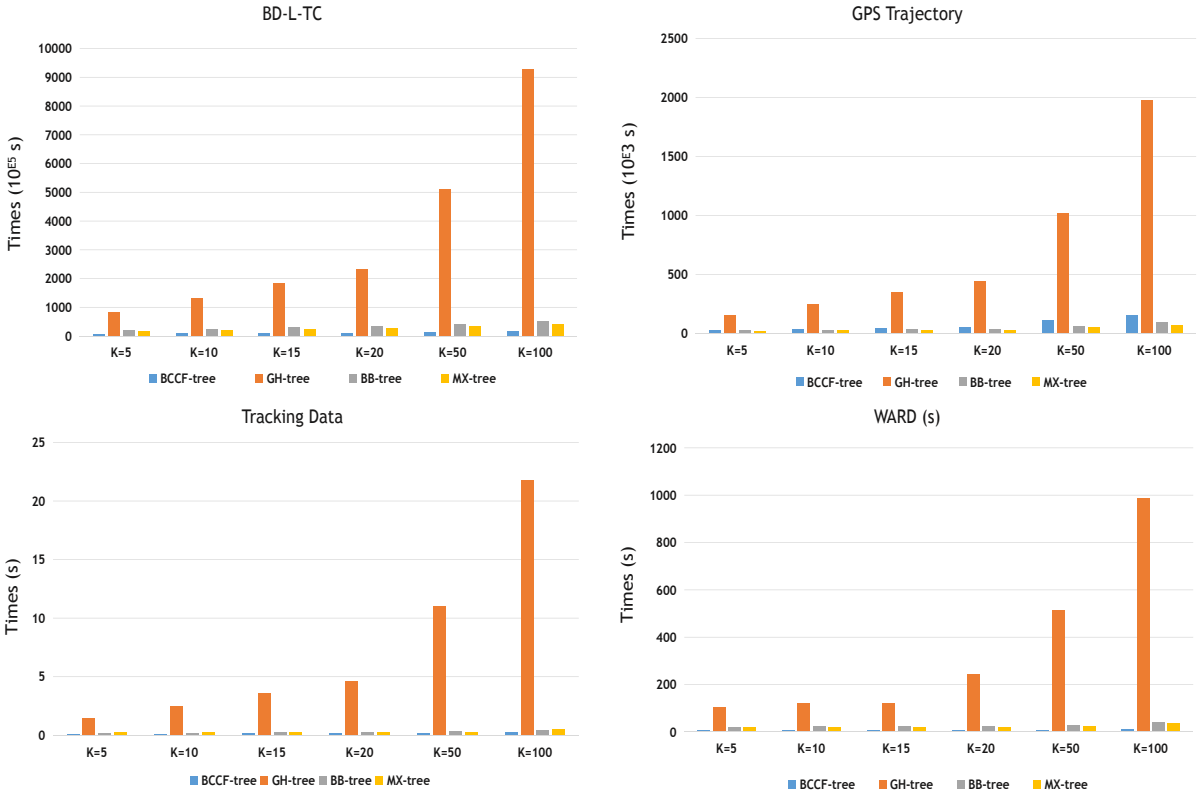


FIGURE 5.13: The kNN search time of BCCF, GH, BB, MX trees

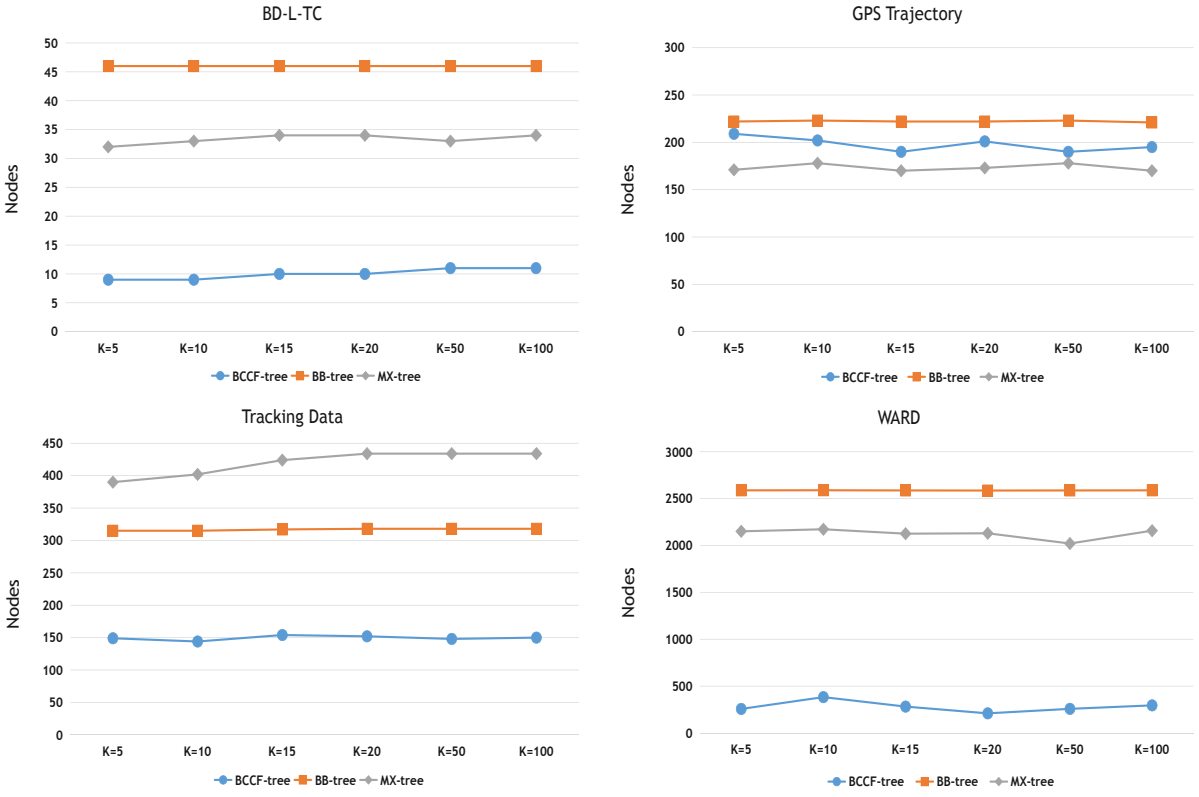


FIGURE 5.14: The average number of leaf nodes visited during the simulation search kNN in BCCF, BB, MX trees

The search process in the BCCF-tree starts from the root of the tree and descends through the internal nodes until it reaches the leaf nodes. Each leaf node has a pointer to another node stored in the cloud containing their real data (see Figure 5.2). So, the communication between these two paradigms (cloud-fog computing) is necessary to retrieve the real data and the number of communications between fog-cloud computing during the search process is exactly the same number of leaf nodes visited. Consequently, Figure 5.14 also shows the number of communications established between the fog and the cloud computing level during the kNN research process.

5.4.5 Evaluation of data distribution

In this section, the k-means algorithm is used to partition data when creating the proposed index tree to cluster similar objects and separate different objects to improve data discovery and retrieval quality. This method is used due to its simplicity, rapid convergence and low complexity.

The k-means algorithm does not have information on the class affiliation of objects. To evaluate the quality of the partitioning of objects by this algorithm, it is necessary to study the degree of homogeneity of all the subclasses (clusters) of the partition. In addition, it is also essential to measure the quality of the separation between the subclasses of the score. The most popular criterion used to evaluate these measures is the inertia criterion. The following is a definition of some of the concepts used in the proposed evaluation.

Centre of Gravity : the centre of gravity of the set of points $N = \{x_i, i = 1, ..., n\}$ is the point G of coordinates $(\bar{x_j}, j = 1, ..., p)$, where :

$$\bar{x_j} = \frac{1}{n} \sum_{i=1}^{n} x_{i,j}$$

Total Inertia : the total inertia of $N = \{x_i, i = 1, ..., n\}$ with the global centre of gravity G is calculated by the following expression:

$$I_{total} = \frac{1}{n} \sum_{i=1}^{n} d^2(x_i, G)$$

or by the sum of two following terms according to Huygens' theorem [62]:

$$I_{total}(N) = I_{intra} + I_{inter}$$

where, I_{intra} corresponds to intra-class inertia, and I_{inter} to inter-class inertia.

Intra-class Inertia : Intra-class inertia is a measure of class compactness. A class is all the more compact when its intra-class inertia is low (i.e. the inertia of a given class is even higher when the points belonging to it are far from its centre). The Intra-class inertia of the set of points $N = \{x_i, i = 1, ..., n\}$ composed of k classes $\{n_j, j = 1, ..., k\}$ with local centres of gravity $\{g_j, j = 1, ..., k\}$ is calculated by the weighted sum of the inertia of each class in relation to their local centres of gravity:

$$I_{intra} = \sum_{j=1}^{k} \frac{n_j}{n} \ (I_j)$$

TABLE 5.5: The exact values of the execution time during the kNN search in the BCCF, GH, BB and MX trees

	BD-L-TC (10^{-5}s)				**GPS Trajectory (10^{-3}s)**				**Tracking Data (s)**				**WARD (s)**			
	BCCF	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**	**BCCF**	**GH**	**BB**	**MX**
k=5	68,68	815,32	196,37	154,12	28,56	151,54	26,12	19,92	0,11	1,42	0,20	0,24	5,93	102,56	20,09	18,94
k=10	81,81	1326,24	246,2	182,85	36,66	249,06	29,53	23,33	0,12	2,48	0,20	0,24	6,69	121,07	21,43	18,94
k=15	96,95	1830,4	301,14	231,71	43,55	345,06	33,07	25,25	0,13	3,56	0,21	0,25	6,80	121,67	22,64	20,05
k=20	104,46	2314,06	349,68	251,91	55,43	442,76	37,15	28,57	0,14	4,63	0,23	0,27	7,03	242,28	23,01	20,25
k=50	125,6	5111,12	422,6	326,94	106,9	1019,57	62,91	48,64	0,17	11,04	0,30	0,29	8,09	514,35	28,61	21,23
k=100	149,74	9277,47	514,17	392,76	149,55	1975,54	93,75	70,09	0,24	21,77	0,42	0,54	11,44	987,68	39,29	35,75

TABLE 5.6: The exact values of the quality parameters of the data distribution

Inertia Value	Intra-class	Inter-class	Total	Partition Ratio
BD-L-TC	0.0012	0.0543	0.0555	0.97
GPS Trajectory	392.3481	33045034	33045426.34	0.99
Tracking Data	7365.6671	212406.52	219772.19	0.96
WARD	42731.0369	1198683.39	1241414.42	0.96

Inter-class Inertia : Inter-class inertia reflects the separability of the different classes. This value is higher when the different local centres of gravity g_j are far from the global centre of gravity G (i.e., well separated classes). The Inter-class inertia of the set of points $N = \{x_i, i = 1, ..., n\}$ composed of k classes $\{n_j, j = 1, ..., k\}$ with local centres of gravity $\{g_j, j = 1, ..., k\}$ is also calculated by the following weighted sum:

$$I_{intra} = \sum_{j=1}^{k} \frac{n_j}{n} \, d^2(g_j, G)$$

The quality rate of a partitioning of an object set is calculated by the following expression:

$$0 \leq \frac{Inter - class \ Inertia}{Total \ Inertia} \leq 1$$

where, the closer the value of this ratio is to 1, the better the distribution is.

Table 5.6 shows the evaluation of the data distribution by the k-means algorithm according to the criteria established above. The results obtained indicate that the k-means algorithm gives positive results for all datasets, where the rate has reached an average value of 0.97. Compared with the data partitioning of the other structures presented in Figure 5.15, BCCF-tree achieves the best data partitioning.

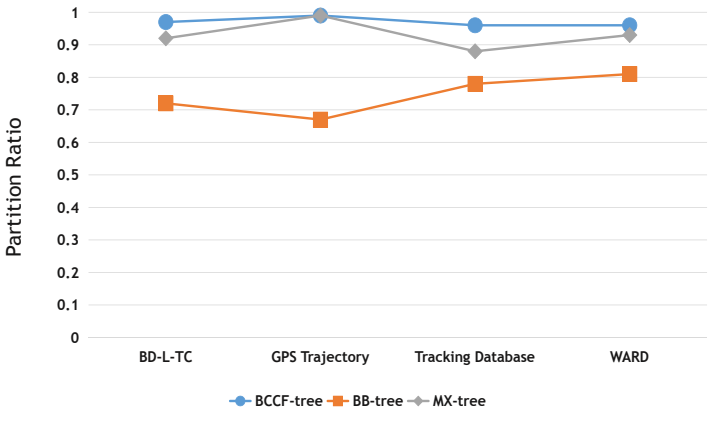


FIGURE 5.15: Data partition rate in the BCCF, BB, and MX trees

5.5 Conclusion

This chapter presented a new indexing structure distributed over IoVT infrastructures to organise the video surveillance system's data in particular and for IoT data in general. This structure is called *B*inary tree based on *C*ontainers at *C*loud-*F*og computing level (BCCF-tree). BCCF-tree is distributed on two levels. At the fog level, which contains the feature vectors indexed in local indexes, and at the cloud level, which contains the indexed feature vectors' real data. BCCF-tree structure composed of three levels. The internal node level contains the pivots; the leaf node level includes a set of data in containers, and the real data container level. The first two levels together form a local index that stores in the Fog nodes. Each leaf node in the local index has a respective container of real data—these containers are stored in the cloud.

The proposed indexing structure is adaptable to our proposed architecture to benefit from their advantages, representing the most powerful and real-time processing capacity provided by fog computing due to its proximity to sensors and the larger storage capacity offered by cloud computing. BCCF-tree is an indexing structure based on the partitioning of data by the k-means clustering algorithm. The use of k-means is to effectively separate objects in the tree nodes, which improves the quality of the search processes. The proposed approach aims to introduce a new distributed indexing structure adapted to the IoT's emerging technologies to enhance the quality of real-time IoT data indexing in general and IoVT data in particular.

CONCLUSION AND PERSPECTIVES

VSS is increasingly receiving a lot of attention as an active area of research. Most of today's systems can process video streams and implement low-level analytical functions. Recently, video surveillance's technical focus has shifted from low-level tasks to analysing more complex scenes to understand behaviours and interpret activities. With the development of computer vision techniques through the introduction of artificial intelligence and distributed computing and distributed intelligence, VSS has become capable of automatically recognising irregular patterns and behaviours in a video and have become adaptable to scene changes and the complexity of the communication network. As a result, VSS has become smarter and more efficient, named "Intelligent Video Surveillance Systems or IVSS". However, the latter requires a flexible and robust infrastructure.

The work described in this thesis revolves around collaborative moving object tracking cameras based on the IoVT paradigms. In what follows, we expose the main contributions made during the thesis period. Then, we present some research directions that deserve to be explored.

In the literature, several paradigms have been used as a video surveillance system infrastructure. Since video surveillance applications are made up of different tasks performed in a bottom-up manner, they are distributed on different paradigms according to the task's needs and the available resources at the card's selected level. Therefore, we have proposed a classification of video surveillance systems' applications into four categories from a low-level paradigm close to the edge network to a high-level paradigm close to the core of the network according to the tasks' needs the resources available in every paradigm.

Based on existing video surveillance systems, we have found that the lack of communication or coordination between cameras causes several problems that negatively influence system performance and object tracking quality. Thus, we have proposed a new distributed and collaborative tracking system, which allows increasing network life, reducing the cost of processing, decreasing communicating data, and improving the quality of tracking based on IoVT computing. The main idea is to elect one camera as a leader among a group of cameras based on overlapping fields of view to ensure tracking. The system can also consider using assistants to enhance efficiency when the leader's vision is not sufficient to follow the object. The other non-selected neighbouring cameras remain inactive to reduce resources consumption.

Our system's coordination mechanism is not applied on all cameras in the system, instead only on sets of cameras that we call clusters or communities. This strategy aims to facilitate the implementation and reduce the coordination mechanism's complexity and restrict the communication area between cameras only at the cluster level. To find the most suitable cameras to cooperate together, we propose two methods of grouping cameras based on overlap surface criteria. The first proposed technique is based on the ascending hierarchical classification (AHC) algorithm. This technique starts with the grouping of strongly overlapping cameras. Nevertheless, this technique has partial knowledge of the network and its state, i.e., it only knows the maximum overlap. The other overlaps are not taken into account (i.e., are entirely neglected). For this purpose, we have proposed a second clustering technique that allows us to regroup not only the two cameras that overlap the most but also all cameras that overlap with as many cameras as possible. To find this type of group, we model the network of cameras with a graph where the vertices represent the cameras and the arcs represent the existence of overlaps. This means that there is an arc between two nodes if and only if the two cameras overlap. Based on this modelling, we observe that the group of cameras we have to find represents a set of nodes (cameras) where they connect two by two. The latter is recognised by a *Clique* in graph theory. To find this type of graph (i.e., Clique), we have used the Bron-karboch Clique search algorithm, which allows us to find the maximum Clique, i.e., the maximum number of overlapping cameras.

Based on the experimental results of our collaborative tracking system, we find that during the re-identification (tagging) process, the system faces the problem of late response to camera requests at the Fog computing level where object data is stored. The problem here is not a latency problem but a sequential search problem in the stored data, which increases dramatically with the system uptime. To remedy this problem, we need to organise the data to find the data requested quickly and efficiently. Consequently, we propose a new and efficient indexing structure based tree to index massive data called BCCF-tree (Binary tree based on containers at the Cloud-Fog computing level). This structure is based on recursive partitioning of space using the k-means clustering algorithm to effectively separate space into nonoverlapping subspace to improve the quality of search and discovery algorithm results. BCCF-tree structure benefits the emerging IoVT computing system, representing the most powerful real-time processing capacity provided by Fog computing due to its proximity to sensors and the largest storage capacity provided by Cloud computing. During our bibliographic study of indexing mechanisms, we proposed a taxonomy for these mechanisms and their various relative structures that allow us and readers to summarise and describe literary works in a simple way.

In conclusion, the creativity of this thesis resides in six points:

1. The classification of video surveillance systems according to the IoVT paradigm(s) used;
2. A new distributed video surveillance system architecture based on the IoVT computing paradigm supports a large-scale multi-camera network;
3. Two new camera grouping techniques based on the FoV overlap area criterion to restrict communication and coordination only at the group level and not for all system cameras;
4. New distributed and collaborative tracking system based on the modern computing paradigm of IoVT to reduce active cameras and increase tracking quality;
5. A new taxonomy of indexing mechanisms and their corresponding structures;
6. A new tree-based data indexing mechanism that supports the proposed architecture.

Despite the efforts made to create a fully distributed intelligent video surveillance system on a large scale, it is still far from the desired perfection. As a perspective of this work, we propose to attain our thesis's final objective, which is developing an efficient approach to analyse suspicious behaviours. Also, several interesting avenues seem to us for the moment to explore, including the problem of grouping cameras. In this context, we propose to improve the clustering method to a dynamic approach that considers the overlapping surfaces and the contextual state of the tracked objects. In other words, camera clustering should be a part of the coordination mechanism and not a preparation for it.

Like other prospects, we plan to exploit UAVs' advantages by combining them with our system to solve the problem of tracking objects in dead zones and propose an extended system where the two systems complement each other, taking into account the heterogeneity of the network and data.

It would be interesting, also, to distribute the indexing structure itself on the available nodes. On the one hand, to efficiently exploit all the resources available in the infrastructure, on the other hand, to facilitate parallelism of search to approximate the search time to logarithmic ($O(log(n))$), which has not yet been reached so far.

BIBLIOGRAPHY

[1] Aalsalem, M. Y., Khan, W. Z., and Dhabbah, K. M. An automated vehicle parking monitoring and management system using anpr cameras. In *2015 17th International Conference on Advanced Communication Technology (ICACT)* (2015), pp. 706–710.

[2] Aazam, M., and Huh, E.-N. Fog computing and smart gateway based communication for cloud of things. In *2014 International Conference on Future Internet of Things and Cloud* (2014), IEEE, pp. 464–470.

[3] Aazam, M., and Huh, E.-N. Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications* (2015), IEEE, pp. 687–694.

[4] Abbasi, A. A., and Younis, M. A survey on clustering algorithms for wireless sensor networks. *Computer communications 30*, 14-15 (2007), 2826–2841.

[5] Abbasifard, M. R., Ghahremani, B., and Naderi, H. A survey on nearest neighbor search methods. *International Journal of Computer Applications 95*, 25 (2014), 39–52.

[6] Abdelaziz, J. *Architectural model for Collaboration in The Internet of Things: a Fog Computing based approach.* PhD thesis, Université du Québec à Chicoutimi, 2018.

[7] Abidi, B. R., Aragam, N. R., Yao, Y., and Abidi, M. A. Survey and analysis of multimodal sensor planning and integration for wide area surveillance. *ACM Computing Surveys (CSUR) 41*, 1 (2009), 1–36.

[8] Abu-Elkheir, M., Hayajneh, M., and Ali, N. A. Data management for the internet of things: Design primitives and solution. *Sensors 13*, 11 (2013), 15582–15612.

[9] Adam, M. S., Anisi, M. H., and Ali, I. Object tracking sensor networks in smart cities: Taxonomy, architecture, applications, research challenges and future directions. *Future Generation Computer Systems 107* (2020), 909–923.

[10] Adil, B., Nadjib, K. M., and Yacine, L. A novel approach for facial expression recognition. In *2019 International Conference on Networking and Advanced Systems (ICNAS)* (2019), IEEE, pp. 1–5.

[11] Adrien, F.-B. Data preprocessing and other improvements for the multi-dimensional ph-index. Master's thesis, ETH Zurich, 2014.

[12] Aghajan, H., and Cavallaro, A. *Multi-Camera Networks: Principles and Applications*. Elsevier Science, 2009.

[13] Aguiló, I., Valverde, L., and Escrig, M. T. *Artificial intelligence research and development*, vol. 100. IOS Press, 2003.

[14] Ahamed, B. B., Ramkumar, T., and Hariharan, S. Data integration progression in large data source using mapping affinity. In *2014 7th International Conference on Advanced Software Engineering and Its Applications* (2014), IEEE, pp. 16–21.

[15] Ahn, H.-K., Mamoulis, N., and Wong, H. M. A survey on multidimensional access methods.

[16] Akiyama, T., Kobayashi, Y., Kishigami, J., and Muto, K. Cnn-based boat detection model for alert system using surveillance video camera. In *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)* (2018), IEEE, pp. 669–670.

[17] Alaei, M., and Barcelo-Ordinas, J. M. A cluster-based scheduling for object detection in wireless multimedia sensor networks. In *Proceedings of the 5th ACM symposium on QoS and security for wireless and mobile networks* (2009), ACM, pp. 50–56.

[18] Alaei, M., and Barcelo-Ordinas, J. M. Mcm: multi-cluster-membership approach for fov-based cluster formation in wireless multimedia sensor networks. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference* (2010), ACM, pp. 1161–1165.

[19] Alaei, M., and Barcelo-Ordinas, J. M. A method for clustering and cooperation in wireless multimedia sensor networks. *Sensors 10*, 4 (2010), 3145–3169.

[20] Alaei, M., and Barcelo-Ordinas, J. M. Node clustering based on overlapping fovs for wireless multimedia sensor networks. In *2010 IEEE Wireless Communication and Networking Conference* (2010), IEEE, pp. 1–6.

[21] Alaei, M., and Barcelo-Ordinas, J. M. A collaborative node management scheme for energy-efficient monitoring in wireless multimedia sensor networks. *Wireless networks 19*, 5 (2013), 639–659.

[22] AlKhamese, A. Y., Shabana, W. R., and Hanafy, I. M. Data security in cloud computing using steganography: A review. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)* (2019), IEEE, pp. 549–558.

[23] Alsmirat, M. A., Jararweh, Y., Obaidat, I., and Gupta, B. B. Internet of surveillance: a cloud supported large-scale wireless surveillance system. *The Journal of Supercomputing 73*, 3 (2017), 973–992.

[24] Alvi, S., Afzal, B., Shah, G., Atzori, L., and Mahmood, W. Internet of multimedia things: Vision and challenges. *Ad Hoc Networks 33* (2015), 87–111.

[25] Amin, A. H. M., Ahmad, N. M., and Ali, A. M. M. Decentralized face recognition scheme for distributed video surveillance in iot-cloud infrastructure. In *2016 IEEE Region 10 Symposium (TENSYMP)* (2016), IEEE, pp. 119–124.

[26] Amrutha, C., Jyotsna, C., and Amudha, J. Deep learning approach for suspicious activity detection from surveillance video. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)* (2020), IEEE, pp. 335–339.

[27] An, J., Chen, Y.-P. P., Xu, Q., and Zhou, X. A new indexing method for high dimensional dataset. In *International Conference on Database Systems for Advanced Applications* (2005), Springer, pp. 385–397.

[28] Antoro, S., Sugeng, K., and Handari, B. Application of bron-kerbosch algorithm in graph clustering using complement matrix. In *AIP Conference Proceedings* (2017), vol. 1862, AIP Publishing, p. 030141.

[29] Antoshenkov, G. Byte-aligned bitmap compression. In *Proceedings DCC'95 Data Compression Conference* (1995), IEEE, p. 476.

[30] Antoshenkov, G., and Ziauddin, M. Query processing and optimization in oracle rdb. *The VLDB Journal 5*, 4 (1996), 229–237.

[31] Arabi, K. Mobile computing opportunities challenges and technology drivers. In *Proc. Design Autom. Conf.* (2014).

[32] Arabi, K. Trends, opportunities and challenges driving architecture and design of next generation mobile computing and iot devices, 2015. Accessed: 2020-09-30.

[33] Arkian, H. R., Diyanat, A., and Pourkhalili, A. Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications. *Journal of Network and Computer Applications 82* (2017), 152–165.

[34] Asare-Bediako, B., Kling, W. L., and Ribeiro, P. F. Multi-agent system architecture for smart home energy management and optimization. In *IEEE PES ISGT Europe 2013* (2013), IEEE, pp. 1–5.

[35] Ashton, K., et al. That 'internet of things' thing. *RFID journal 22*, 7 (2009), 97–114.

[36] Athanassoulis, M., Yan, Z., and Idreos, S. Upbit: Scalable in-memory updatable bitmap indexing. In *Proceedings of the 2016 International Conference on Management of Data* (2016), ACM, pp. 1319–1332.

[37] Atlam, H., Walters, R., Wills, G., et al. Internet of things: state-of-the-art, challenges, applications, and open issues. *International Journal of Intelligent Computing Research (IJICR) 9*, 3 (2018), 928–938.

[38] Atlam, H. F., Alenezi, A., Walters, R. J., Wills, G. B., and Daniel, J. Developing an adaptive risk-based access control model for the internet of things. In *2017 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and ieee smart data (SmartData)* (2017), IEEE, pp. 655–661.

[39] Atlam, H. F., Walters, R. J., and Wills, G. B. Fog computing and the internet of things: a review. *big data and cognitive computing 2*, 2 (2018), 10.

[40] Atzori, L., Iera, A., and Morabito, G. The internet of things: A survey. *Computer networks 54*, 15 (2010), 2787–2805.

[41] Avola, D., Foresti, G. L., Martinel, N., Micheloni, C., Pannone, D., and Piciarelli, C. Aerial video surveillance system for small-scale uav environment monitoring. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (2017), IEEE, pp. 1–6.

[42] Bachmann, J. P. The superm-tree: Indexing metric spaces with sized objects. *arXiv preprint arXiv:1901.11453* (2019).

[43] Balasubramanian, B., Durai, K., Sathyanarayanan, J., and Muthukumarasamy, S. Tree based fast similarity query search indexing on outsourced cloud data streams. *INTERNATIONAL ARAB JOURNAL OF INFORMATION TECHNOLOGY 16*, 5 (2019), 871–878.

[44] Banafa, A. What is fog computing?, 2015.

[45] Barhoush, M. M., AlSobeh, A. M., and Al Rawashdeh, A. A survey on parallel join algorithms using mapreduce on hadoop. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)* (2019), IEEE, pp. 381–388.

[46] Bayer, R. The universal b-tree for multidimensional indexing: General concepts. In *International Conference on Worldwide Computing and Its Applications* (1997), Springer, pp. 198–209.

[47] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. The r*-tree: an efficient and robust access method for points and rectangles. In *Acm Sigmod Record* (1990), vol. 19, Acm, pp. 322–331.

[48] Behera, R. K., Kharade, P., Yerva, S., Dhane, P., Jain, A., and Kutty, K. Multi-camera based surveillance system. In *2012 World Congress on Information and Communication Technologies* (2012), IEEE, pp. 102–108.

[49] Ben Hamida, A., Koubaa, M., Nicolas, H., and Amar, C. B. Video surveillance system based on a scalable application-oriented architecture. *Multimedia Tools and Applications 75*, 24 (2016), 17187–17213.

[50] Benrazek, A.-E., Brahim, F., and Muhammet, K. Efficient camera clustering method based on overlapping fovs for wmsns. *International Journal of Informatics and Applied Mathematics 1*, 1 (2018), 10–23.

[51] Benrazek, A.-E., Farou, B., Seridi, H., Kouahla, Z., and Kurulay, M. Ascending hierarchical classification for camera clustering based on fov overlaps for wmsn. *IET Wireless Sensor Systems 9*, 6 (2019), 382–388.

[52] Benrazek, A.-E., Kouahla, Z., Farou, B., Ferrag, M. A., Seridi, H., and Kurulay, M. An efficient indexing for internet of things massive data based on cloud-fog computing. *Transactions on Emerging Telecommunications Technologies 31*, 3 (2020), e3868.

[53] Bentley, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 9 (1975), 509–517.

[54] Bentley, J. L. Multidimensional binary search trees in database applications. *IEEE Transactions on Software Engineering*, 4 (1979), 333–340.

[55] Berchtold, S., Böhm, C., and Kriegal, H.-P. The pyramid-technique: towards breaking the curse of dimensionality. In *ACM SIGMOD Record* (1998), vol. 27, ACM, pp. 142–153.

[56] Berchtold, S., Keim, D. A., and Kriegel, H.-P. The x-tree: An index structure for high-dimensional data. In *Proceedings of the 22th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1996), VLDB '96, Morgan Kaufmann Publishers Inc., pp. 28–39.

[57] Berg, M. d., Cheong, O., Kreveld, M. v., and Overmars, M. *Computational Geometry: Algorithms and Applications*, 3rd ed. ed. Springer-Verlag TELOS, Santa Clara, CA, USA, 2008.

[58] Berry, G. *Real time programming: Special purpose or general purpose languages.* PhD thesis, INRIA, 1989.

[59] Beutel, A., Kraska, T., Chi, E., Dean, J., and Polyzotis, N. A machine learning approach to databases indexes. In *ML Systems Workshop* (2017), NIPS.

[60] Bhuvana, V. P., Schranz, M., Regazzoni, C. S., Rinner, B., Tonello, A. M., and Huemer, M. Multi-camera object tracking using surprisal observations in visual sensor networks. *EURASIP Journal on Advances in Signal Processing 2016*, 1 (2016), 50.

[61] Bisagno, N., Xamin, A., De Natale, F., Conci, N., and Rinner, B. Dynamic camera reconfiguration with reinforcement learning and stochastic methods for crowd surveillance. *Sensors 20*, 17 (2020), 4691.

[62] Boes, D. C., Graybill, F. A., and Mood, A. M. *Introduction to the Theory of Statistics.* McGraw-Hill International Book Company, 1982.

[63] Böhm, C., Berchtold, S., and Keim, D. A. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys (CSUR) 33*, 3 (2001), 322–373.

[64] Böhm, C., Berchtold, S., Kriegel, H.-P., and Michel, U. Multidimensional index structures in relational databases. *Journal of Intelligent Information Systems 15*, 1 (2000), 51–70.

[65] Bonomi, F. Connected vehicles, the internet of things, and fog computing. In *The eighth ACM international workshop on vehicular inter-networking (VANET), Las Vegas, USA* (2011), pp. 13–15.

[66] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* (2012), ACM, pp. 13–16.

[67] Booch, G., Maksimchuk, R. A., Engle, M. W., Young, B. J., Connallen, J., and Houston, K. A. Object-oriented analysis and design with applications. *ACM SIGSOFT software engineering notes 33*, 5 (2008), 29–29.

[68] Bovik, A. C. *The essential guide to image processing.* Academic Press, 2009.

[69] Bozkaya, T., and Ozsoyoglu, M. Distance-based indexing for high-dimensional metric spaces. In *ACM SIGMOD Record* (1997), vol. 26, ACM, pp. 357–368.

[70] Bramberger, M., Doblander, A., Maier, A., Rinner, B., and Schwabach, H. Distributed embedded smart cameras for surveillance applications. *Computer 39*, 2 (2006), 68–75.

[71] Bramberger, M., Quaritsch, M., Winkler, T., Rinner, B., and Schwabach, H. Integrating multi-camera tracking into a dynamic task allocation system for smart cameras. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.* (2005), IEEE, pp. 474–479.

[72] Brin, S. Near neighbor search in large metric spaces.

[73] Brinis, S., Traina, C., and Traina, A. J. Hollow-tree: a metric access method for data with missing values. *Journal of Intelligent Information Systems* (2019), 1–28.

[74] Bron, C., and Kerbosch, J. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM 16*, 9 (1973), 575–577.

[75] Bruynooghe, M. Classification ascendante hiérarchique des grands ensembles de données: un algorithme rapide fondé sur la construction des voisinages réductibles. *Cahiers de l'analyse des données 3*, 1 (1978), 7–33.

[76] Bunz, M., and Meikle, G. *The internet of things.* John Wiley & Sons, 2017.

[77] Burns, A., and Wellings, A. J. *Real-time systems and programming languages: Ada 95, real-time Java, and real-time POSIX.* Pearson Education, 2001.

[78] Buyya, R., Broberg, J., and Goscinski, A. M. *Cloud computing: Principles and paradigms*, vol. 87. John Wiley & Sons, 2010.

[79] Buyya, R., and Dastjerdi, A. V. *Internet of Things: Principles and paradigms.* Elsevier, 2016.

[80] Buyya, R., and Srirama, S. N. *Fog and edge computing: principles and paradigms.* John Wiley & Sons, 2019.

[81] Cakir, F., He, K., Bargal, S. A., and Sclaroff, S. Hashing with mutual information. *IEEE transactions on pattern analysis and machine intelligence* (2019).

[82] Carélo, C. C. M., Pola, I. R. V., Ciferri, R. R., Traina, A. J. M., Jr., C. T., and de Aguiar Ciferri, C. D. Slicing the metric space to provide quick indexing of complex data in the main memory. *Inf. Syst 36* (2011), 79–98.

[83] Casagras, E. Casagras final report: Rfid and the inclusive model for the internet of things. *EU FP7 Project CASAGRAS* (2009).

[84] Castañeda, J. N., Jelaca, V., Frías, A., Pizurica, A., Philips, W., Cabrera, R. R., and Tuytelaars, T. Non-overlapping multi-camera detection and tracking of vehicles in tunnel surveillance. In *2011 International Conference on Digital Image Computing: Techniques and Applications* (2011), IEEE, pp. 591–596.

[85] Castro, F. M., Delgado-Escaño, R., Guil, N., and Marín-Jiménez, M. J. A weakly-supervised approach for discovering common objects in airport video surveillance footage. 296–308.

[86] Chamasemani, F. F., and Affendey, L. S. Systematic review and classification on video surveillance systems. *International Journal of Information Technology and Computer Science (IJITCS) 5*, 7 (2013), 87.

[87] Chambi, S., Lemire, D., Kaser, O., and Godin, R. Better bitmap performance with roaring bitmaps. *Software: practice and experience 46*, 5 (2016), 709–719.

[88] Chan, C.-Y., and Ioannidis, Y. E. Bitmap index design and evaluation. In *ACM SIGMOD Record* (1998), vol. 27, ACM, pp. 355–366.

[89] Chandrasekaran, K. *Essentials of cloud computing*. CrC Press, 2014.

[90] Chang, C., Srirama, S. N., and Buyya, R. Internet of things (iot) and new computing paradigms. *Fog and edge computing: principles and paradigms 6* (2019), 1–23.

[91] Chang, C.-C., and Tsai, J. Distributed collaborative surveillance system based on leader election protocols. *IET Wireless Sensor Systems 6*, 6 (2016), 198–205.

[92] Chang, J., Chen, Z., Zheng, W., Cao, J., Wen, Y., Peng, G., and Huang, W.-L. Splwah: A bitmap index compression scheme for searching in archival internet traffic. In *2015 IEEE International Conference on Communications (ICC)* (2015), IEEE, pp. 7089–7094.

[93] Chaurasiya, S. K., Mondal, J., and Dutta, S. Field-of-view based hierarchical clustering to prolong network lifetime of wmsn with obstacles. In *Electronics, Communication and Computational Engineering (ICECCE), 2014 International Conference on* (2014), IEEE, pp. 72–77.

[94] Chávez, E., Navarro, G., Baeza-Yates, R., and Marroquín, J. L. Searching in metric spaces. *ACM computing surveys (CSUR) 33*, 3 (2001), 273–321.

[95] Chee, B. J., and Franklin Jr, C. *Cloud computing: technologies and strategies of the ubiquitous data center*. CRC Press, 2010.

[96] Chen, G., Yang, K., Chen, L., Gao, Y., Zheng, B., and Chen, C. Metric similarity joins using mapreduce. *IEEE Transactions on Knowledge and Data Engineering 29*, 3 (2016), 656–669.

[97] Chen, L., Gao, Y., Li, X., Jensen, C. S., and Chen, G. Efficient metric indexing for similarity search. In *2015 IEEE 31st International Conference on Data Engineering* (2015), IEEE, pp. 591–602.

[98] Chen, L., Gao, Y., Li, X., Jensen, C. S., and Chen, G. Efficient metric indexing for similarity search and similarity joins. *IEEE Transactions on Knowledge and Data Engineering 29*, 3 (2015), 556–571.

[99] Chen, N., Chen, Y., You, Y., Ling, H., Liang, P., and Zimmermann, R. Dynamic urban surveillance video stream processing using fog computing. In *2016 IEEE second international conference on multimedia big data (BigMM)* (2016), IEEE, pp. 105–112.

[100] Chen, S., Zhang, T., and Shi, W. Fog computing. *IEEE Internet Computing 21*, 2 (2017), 4–6.

[101] Cheng, H., Yang, W., Tang, R., Mao, J., Luo, Q., Li, C., and Wang, A. Distributed indexes design to accelerate similarity based images retrieval in airport video monitoring systems. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)* (2015), IEEE, pp. 1908–1912.

[102] Chi, L., and Zhu, X. Hashing techniques: A survey and taxonomy. *ACM Computing Surveys (CSUR) 50*, 1 (2017), 11.

[103] Chiang, M., and Zhang, T. Fog and iot: An overview of research opportunities. *IEEE Internet of Things Journal 3*, 6 (2016), 854–864.

[104] Ciaccia, P., Patella, M., Rabitti, F., and Zezula, P. Indexing metric spaces with m-tree. In *SEBD* (1997), vol. 97, pp. 67–86.

[105] Ciaccia, P., Patella, M., and Zezula, P. M-tree: An efficient access method for similarity search in metric spaces. *Proceedings of the 23rd VLDB International Conference* (1997), 426–435.

[106] Ciaccia, P., Patella, M., and Zezula, P. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd VLDB conference, Athens, Greece* (1997), Citeseer, pp. 426–435.

[107] Cisco. The internet of everything: Global public sector economic analysis, 2013.

[108] Cisco. The internet of things reference model, 2014.

[109] Clough, B. T. Metrics, schmetrics! how the heck do you determine a uav's autonomy anyway. Tech. rep., Air Force Research Lab Wright-Patterson AFB OH, 2002.

[110] Colantonio, A., and Di Pietro, R. Concise: Compressed 'n'composable integer set. *Information Processing Letters 110*, 16 (2010), 644–650.

[111] Computing, F. the internet of things: Extend the cloud to where the things are. *Cisco White Paper* (2015).

[112] Consortium, O., et al. Openfog reference architecture for fog computing. *Architecture Working Group* (2017), 1–162.

[113] Costa, D. G., Silva, I., Guedes, L. A., Vasques, F., and Portugal, P. Optimal sensing redundancy for multiple perspectives of targets in wireless visual sensor networks. In *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on* (2015), IEEE, pp. 185–190.

[114] Costa, F. *ND-Tree: Multidimensional Indexing Structure*. Novas Edições Acadêmicas, 2017.

[115] Cruz, M. O., Macedo, H., and Guimaraes, A. Grouping similar trajectories for carpooling purposes. In *2015 Brazilian Conference on Intelligent Systems (BRACIS)* (2015), IEEE, pp. 234–239.

[116] Dahir, H., and Dry, B. *People, Processes, Services, and Things: Using Services Innovation to Enable the Internet of Everything*. Business Expert Press, 2015.

[117] Dastjerdi, A. V., Gupta, H., Calheiros, R. N., Ghosh, S. K., and Buyya, R. Fog computing: Principles, architectures, and applications. In *Internet of things*. Elsevier, 2016, pp. 61–75.

[118] David, R., John, G., and John, R. The digitization of the world: From edge to core (idc white paper# us44413318), 2018.

[119] De Donno, M., Tange, K., and Dragoni, N. Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog. *Ieee Access 7* (2019), 150936–150948.

[120] De Rham, C. La classification hiérarchique ascendante selon la méthode des voisins réciproques. *Cahiers de l'analyse des données 5*, 2 (1980), 135–144.

[121] Deliège, F., and Pedersen, T. B. Position list word aligned hybrid: optimizing space and performance for compressed bitmaps. In *Proceedings of the 13th international conference on Extending Database Technology* (2010), ACM, pp. 228–239.

[122] Desai, M., Mehta, R. G., and Rana, D. P. A survey on techniques for indexing and hashing in big data. In *2018 4th International Conference on Computing Communication and Automation (ICCCA)* (2018), IEEE, pp. 1–6.

[123] Detmold, H., van den Hengel, A., Dick, A., Falkner, K., Munro, D. S., and Morrison, R. Middleware for distributed video surveillance. *IEEE Distributed Systems Online 9*, 2 (2008), 1–1.

[124] Dijkstra, E. W. Hierarchical ordering of sequential processes. In *The origin of concurrent programming*. Springer, 1971, pp. 198–227.

[125] Diratie, E. D., and Al Agha, K. Hybrid internet of things network for energy efficient video surveillance. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)* (2020), IEEE, pp. 1–6.

[126] Dohnal, V. *Indexing structures for searching in metric spaces*. PhD thesis, Masaryk University, 2004.

[127] Doja, M., Jain, S., and Alam, M. A. Sas: Implementation of scaled association rules on spatial multidimensional quantitative dataset. *International Journal of Advanced Computer Science and Applications 3*, 9 (2012), 130–135.

[128] Dolatshah, M., Hadian, A., and Minaei-Bidgoli, B. Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces. *arXiv preprint arXiv:1511.00628* (2015).

[129] Dong, W., Wang, Z., Josephson, W., Charikar, M., and Li, K. Modeling lsh for performance tuning. In *Proceedings of the 17th ACM conference on Information and knowledge management* (2008), ACM, pp. 669–678.

[130] Dong, Y., He, J., Yao, S., and Zhou, W. The skip-octree: a dynamic cloud storage index framework for multidimensional big data systems. *International Journal of Web Engineering and Technology 10*, 4 (2015), 393–407.

[131] Dong, Y., Indyk, P., Razenshteyn, I., and Wagner, T. Learning sublinear-time indexing for nearest neighbor search. *CoRR abs/1901.08544* (2019).

[132] D'Orazio, T., and Guaragnella, C. A survey of automatic event detection in multi-camera third generation surveillance systems. *International Journal of Pattern Recognition and Artificial Intelligence 29*, 01 (2015), 1555001.

[133] Durcevic, S. 10 cloud computing risks & challenges businesses are facing in these days, 2019.

[134] Elloy, J.-P. Les contraintes du temps réel dans les systèmes industriels répartis. *Revue générale de l'électricité (Paris)*, 2 (1991), 26–30.

[135] Evans, D. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper 1*, 2011 (2011), 1–11.

[136] Faloutsos, C., and Lin, K.-I. *FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets*, vol. 24. ACM, 1995.

[137] Fang, W., Ding, L., Love, P. E., Luo, H., Li, H., Pena-Mora, F., Zhong, B., and Zhou, C. Computer vision applications in construction safety assurance. *Automation in Construction 110* (2020), 103013.

[138] Farou, B., Kouahla, M. N., Seridi, H., and Akdag, H. Efficient local monitoring approach for the task of background subtraction. *Engineering Applications of Artificial Intelligence 64* (2017), 1–12.

[139] Farou, B., Seridi, H., and Akdag, H. A new approach for the extraction of moving objects. In *Modeling Approaches and Algorithms for Advanced Computer Applications*. Springer, 2013, pp. 27–36.

[140] Farou, B., Seridi, H., and Akdag, H. Improved gaussian mixture model with background spotter for the extraction of moving objects. *Int. Arab J. Inf. Technol. 13*, 6A (2016), 807–816.

[141] Fathy, Y., Barnaghi, P., and Tafazolli, R. Large-scale indexing, discovery, and ranking for the internet of things (iot). *ACM Computing Surveys (CSUR) 51*, 2 (2018), 1–53.

[142] Fedorov, A., Nikolskaia, K., Ivanov, S., Shepelev, V., and Minbaleev, A. Traffic flow estimation with data from a video surveillance camera. *Journal of Big Data 6*, 1 (2019), 73.

[143] Feng, C., Li, C.-D., and Li, R. Indexing techniques of distributed ordered tables: A survey and analysis. *Journal of Computer Science and Technology 33*, 1 (2018), 169–189.

[144] Fenk, R. The bub-tree. In *VLDB'02, Proceedings of 28th International Conference on Very Large Data Bases* (2002), Citeseer.

[145] Ferrag, M. A., Maglaras, L. A., Janicke, H., and Jiang, J. A survey on privacy-preserving schemes for smart grid communications. *arXiv preprint arXiv:1611.07722* (2016).

[146] Finkel, R. A., and Bentley, J. L. Quad trees a data structure for retrieval on composite keys. *Acta informatica 4*, 1 (1974), 1–9.

[147] Foggia, P., Saggese, A., and Vento, M. Real-time fire detection for video-surveillance applications using a combination of experts based on color, shape, and motion. *IEEE TRANSACTIONS on circuits and systems for video technology 25*, 9 (2015), 1545–1556.

[148] Friedman, J. H., Bentley, J. L., and Finkel, R. A. An algorithm for finding best matches in logarithmic time. *ACM Trans. Math. Software 3*, SLAC-PUB-1549-REV. 2 (1976), 209–226.

[149] Fusco, F., Stoecklin, M. P., and Vlachos, M. Net-fli: on-the-fly compression, archiving and indexing of streaming network traffic. *Proceedings of the VLDB Endowment 3*, 1-2 (2010), 1382–1393.

[150] Gaede, V., and Günther, O. Multidimensional access methods. *ACM Computing Surveys (CSUR) 30*, 2 (1998), 170–231.

[151] Gao, Z., Xu, C., Zhang, H., Li, S., and de Albuquerque, V. H. Trustful internet of surveillance things based on deeply represented visual co-saliency detection. *IEEE Internet of Things Journal 7*, 5 (2020), 4092–4100.

[152] Germann, U., Joanis, E., and Larkin, S. Tightly packed tries: How to fit large models into memory, and make them load fast, too. In *Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing* (2009), Association for Computational Linguistics, pp. 31–39.

[153] Gershenfeld, N. *When things start to think*. Macmillan, 1999.

[154] Gharaibeh, A., Salahuddin, M. A., Hussini, S. J., Khreishah, A., Khalil, I., Guizani, M., and Al-Fuqaha, A. Smart cities: A survey on data management, security, and enabling technologies. *IEEE Communications Surveys & Tutorials 19*, 4 (2017), 2456–2501.

[155] Gherbi, C., Aliouat, Z., and Benmohammed, M. A survey on clustering routing protocols in wireless sensor networks. *Sensor Review 37*, 1 (2017), 12–25.

[156] Ghilic-Micu, B., Mircea, M., and Stoica, M. Qaaas in a cloud iot ecosystem. *Informatica Economica 21*, 4 (2017), 5–14.

[157] Gionis, A., Indyk, P., Motwani, R., et al. Similarity search in high dimensions via hashing. In *Vldb* (1999), vol. 99, pp. 518–529.

[158] Giusto, D., Iera, A., Morabito, G., and Atzori, L. *The Internet of Things: 20th Tyrrhenian Workshop on Digital Communications*. Springer New York, 2010.

[159] Gomez, C., Chessa, S., Fleury, A., Roussos, G., and Preuveneers, D. Internet of things for enabling smart environments: A technology-centric perspective. *Journal of Ambient Intelligence and Smart Environments 11*, 1 (2019), 23–43.

[160] Gouaillier, V. La vidéosurveillance intelligente: promesses et défis. *Rapport technique, TechnoPole Defense and Security, CRIM* (2009).

[161] Gouaillier, V., and Fleurant, A.-E. Intelligent video surveillance: Promises and challenges. *Technological and commercial intelligence report, CRIM and Technôpole Defence and Security 456* (2009), 468.

[162] Gowsikhaa, D., Manjunath, and Abirami, S. Suspicious human activity detection from surveillance videos. *International Journal on Internet & Distributed Computing Systems 2*, 2 (2012), 141–148.

[163] Griffith, E., et al. What is cloud computing. *Retrieved from PC Mag: http://au. pcmag. com/networking-communications-software-products/29902/feature/what-is-cloud-computing* (2016).

[164] Gu, J., Su, T., Wang, Q., Du, X., and Guizani, M. Multiple moving targets surveillance based on a cooperative network for multi-uav. *IEEE Communications Magazine 56*, 4 (2018), 82–89.

[165] Gui, J., Liu, T., Sun, Z., Tao, D., and Tan, T. Fast supervised discrete hashing. *IEEE transactions on pattern analysis and machine intelligence 40*, 2 (2017), 490–496.

[166] Gulve, S. P., Khoje, S. A., and Pardeshi, P. Implementation of iot-based smart video surveillance system. In *Computational Intelligence in Data Mining*. Springer Singapore, Singapore, 2017, pp. 771–780.

[167] Gungor, V. C., Hancke, G. P., et al. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *IEEE Trans. Industrial Electronics 56*, 10 (2009), 4258–4265.

[168] Günnemann, S., Kremer, H., Lenhard, D., and Seidl, T. Subspace clustering for indexing high dimensional data: a main memory index based on local reductions and individual multi-representations. In *Proceedings of the 14th International Conference on Extending Database Technology* (2011), ACM, pp. 237–248.

[169] Gupta, H., Chakraborty, S., Ghosh, S. K., and Buyya, R. Fog computing in 5g networks: an application perspective. *Cloud and Fog Computing in 5G Mobile Networks: Emerging Advances and Applications; IET Book: Wales, UK* (2017), 23–56.

[170] Guttman, A. *R-trees: A dynamic index structure for spatial searching*, vol. 14. ACM, 1984.

[171] Guzun, G., Canahuate, G., Chiu, D., and Sawin, J. A tunable compression framework for bitmap indices. In *2014 IEEE 30th International Conference on Data Engineering* (2014), IEEE, pp. 484–495.

[172] Hamida, A. B. *Vers une nouvelle architecture de videosurveillance basée sur la scalabilité orientée vers l'application*. PhD thesis, Université de Bordeaux, 2016.

[173] Han, J., Zhang, D., Cheng, G., Liu, N., and Xu, D. Advanced deep-learning techniques for salient and category-specific object detection: a survey. *IEEE Signal Processing Magazine 35*, 1 (2018), 84–100.

[174] Hanyf, Y., and Silkan, H. A queries-based structure for similarity searching in static and dynamic metric spaces. *Journal of King Saud University-Computer and Information Sciences* (2018).

[175] Harel, D., and Pnueli, A. On the development of reactive systems. In *Logics and models of concurrent systems*. Springer, 1985, pp. 477–498.

[176] Harvey, N. J., Dunagan, J., Jones, M., Saroiu, S., Theimer, M., and Wolman, A. Skipnet: A scalable overlay network with practical locality properties.

[177] Hashemzadeh, M., and Zademehdi, A. Fire detection for video surveillance applications using ica k-medoids-based color model and efficient spatio-temporal visual features. *Expert Systems with Applications 130* (2019), 60–78.

[178] Hassan, Q. F., Madani, S. A., et al. *Internet of things: Challenges, advances, and applications*. Chapman and Hall/CRC, 2017.

[179] He, J., Zhang, Y., Zhou, M., and Han, Y. The design and implementation of real-time bus monitoring based on 4g mobile internet. In *2015 4th National Conference on Electrical, Electronics and Computer Engineering* (2015), Atlantis Press.

[180] He, K., Cakir, F., Adel Bargal, S., and Sclaroff, S. Hashing as tie-aware learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 4023–4032.

[181] He, Y., Wei, X., Hong, X., Shi, W., and Gong, Y. Multi-target multi-camera tracking by tracklet-to-target assignment. *IEEE Transactions on Image Processing 29* (2020), 5191–5205.

[182] Hengstler, S., Prashanth, D., Fong, S., and Aghajan, H. Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In *Proceedings of the 6th international conference on Information processing in sensor networks* (2007), pp. 360–369.

[183] Heo, J.-P., Lee, Y., He, J., Chang, S.-F., and Yoon, S.-E. Spherical hashing. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), IEEE, pp. 2957–2964.

[184] Hewitt, C. Viewing control structures as patterns of passing messages. *Artificial intelligence 8*, 3 (1977), 323–364.

[185] Hewitt, C. Orgs for scalable, robust, privacy-friendly client cloud computing. *IEEE internet computing 12*, 5 (2008), 96–99.

[186] Himmel, A.-S., Molter, H., Niedermeier, R., and Sorge, M. Enumerating maximal cliques in temporal graphs. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (2016), IEEE, pp. 337–344.

[187] Hu, P., Dhelim, S., Ning, H., and Qiu, T. Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of network and computer applications 98* (2017), 27–42.

[188] Hu, W., Huang, Y., Wei, L., Zhang, F., and Li, H. Deep convolutional neural networks for hyperspectral image classification. *Journal of Sensors 2015* (2015).

[189] IBM Cloud, E. Learn what a public cloud is and what it offers compared to—or together with—private cloud and hybrid cloud computing models, 2020.

[190] IEEE, I. I. Towards a definition of the internet of things (iot). *Revision-1, on-line: http://iot. ieee. org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15. pdf. 27*, 2017 (2015), 479–501.

[191] Iorga, M., Feldman, L., Barton, R., Martin, M., Goren, N., and Mahmoudi, C. The nist definition of fog computing. Tech. rep., National Institute of Standards and Technology, 2017.

[192] Iorga, M., Feldman, L., Barton, R., Martin, M., Goren, N., and Mahmoudi, C. Fog computing conceptual model.

[193] Irie, G., Li, Z., Wu, X.-M., and Chang, S.-F. Locally linear hashing for extracting non-linear manifolds. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 2115–2122.

[194] Ismail, Y., Hammad, M., and El-Medany, W. Homeland security video surveillance system for smart cities. In *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)* (2018), IEEE, pp. 1–4.

[195] ITU, I. T. U. The internet of things—itu internet reports, 2005.

[196] Jalew, E. A. *Fog Computing based traffic Safety for Connected Vulnerable Road Users.* PhD thesis, Bourgogne Franche-Comté, 2019.

[197] Jang, H.-J., Kim, B., and Jung, S.-Y. k-nearest reliable neighbor search in crowdsourced lbss. *International Journal of Communication Systems* (2019), e4097.

[198] Jayashree, L. S., and Selvakumar, G. *Cloud Solutions for IoT*. Springer International Publishing, Cham, 2020.

[199] Ji, J., Li, J., Yan, S., Zhang, B., and Tian, Q. Super-bit locality-sensitive hashing. In *Advances in Neural Information Processing Systems* (2012), pp. 108–116.

[200] Jiang, Q.-Y., and Li, W.-J. Scalable graph hashing with feature transformation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence* (2015).

[201] Jiang, Q.-Y., and Li, W.-J. Asymmetric Deep Supervised Hashing. *arXiv e-prints* (Jul 2017), arXiv:1707.08325.

[202] Jin, P., and Song, Q. A novel index structure r* q-tree based on lazy splitting and clustering. In *2011 IEEE International Conference on Computer Science and Automation Engineering* (2011), vol. 3, IEEE, pp. 405–407.

[203] Jin, S., Kim, O., and Feng, W. Mx-tree: A double hierarchical metric index with overlap reduction. In *International Conference on Computational Science and Its Applications* (2013), Springer, pp. 574–589.

[204] Jo, B., and Jung, S. Quadrant-based minimum bounding rectangle-tree indexing method for similarity queries over big spatial data in hbase. *Sensors 18*, 9 (2018), 3032.

[205] Jung, K., Lee, J.-Y., and Jeong, H.-Y. Improving adaptive cluster head selection of teen protocol using fuzzy logic for wmsn. *Multimedia Tools and Applications 76*, 17 (2017), 18175–18190.

[206] Kalantari, I., and McDonald, G. A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, 5 (1983), 631–634.

[207] Kamel, I., and Faloutsos, C. Hilbert r-tree: An improved r-tree using fractals. Tech. rep., 1993.

[208] Kang, W.-C., Li, W.-J., and Zhou, Z.-H. Column sampling based discrete supervised hashing. In *Thirtieth AAAI conference on artificial intelligence* (2016).

[209] Katayama, N. S. i. satoh," the sr-tree: an index structure for highdimensional nearest neighbor queries,". In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*.

[210] Katayama, N., and Satoh, S. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *ACM Sigmod Record* (1997), vol. 26, ACM, pp. 369–380.

[211] KEAWPIBAL, N., PREECHAVEERAKUL, L., and VANICHAYOBON, S. Hybix: A novel encoding bitmap index for space-and time-efficient query processing. *Turkish Journal of Electrical Engineering & Computer Sciences 27* (2019), 1504–1522.

[212] Kelley, J. L. *General topology*. Courier Dover Publications, 2017.

[213] Kelly, R. Internet of things data to top 1.6 zettabytes by 2020, 2015.

[214] Khan, M., Khan, S., and Zomaya, A. *Big Data-Enabled Internet of Things*. Computing and Networks. Institution of Engineering & Technology, 2020.

[215] Kheirkhah, M. M., and Khansari, M. Clustering wireless camera sensor networks based on overlapped region detection. In *Telecommunications (IST), 2014 7th International Symposium on* (2014), IEEE, pp. 712–719.

[216] Kim, I. S., Choi, H. S., Yi, K. M., Choi, J. Y., and Kong, S. G. Intelligent visual surveillance—a survey. *International Journal of Control, Automation and Systems 8*, 5 (2010), 926–939.

[217] Kim, S., and Choi, S. Semi-supervised discriminant hashing. In *2011 IEEE 11th International Conference on Data Mining* (2011), IEEE, pp. 1122–1127.

[218] Kim, S., Lee, J., Satti, S. R., and Moon, B. Sbh: Super byte-aligned hybrid bitmap compression. *Information Systems 62* (2016), 155–168.

[219] Kouahla, Z. *Indexation dans les espaces métriques Index arborescent et parallélisation.* PhD thesis, Université de Nantes, 2013.

[220] Kouahla, Z., and Anjum, A. A parallel implementation of ghb tree. In *IFIP International Conference on Computational Intelligence and Its Applications* (2018), Springer, pp. 47–55.

[221] Kouahla, Z., Anjum, A., Akram, S., Saba, T., and Martinez, J. Xm-tree: data driven computational model by using metric extended nodes with non-overlapping in high-dimensional metric spaces. *Computational and Mathematical Organization Theory 25*, 2 (2019), 196–223.

[222] Kouahla, Z., and Martinez, J. A new intersection tree for content-based image retrieval. In *2012 10th International Workshop on Content-Based Multimedia Indexing (CBMI)* (2012), IEEE, pp. 1–6.

[223] Koubaa, A., and Qureshi, B. Dronetrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the internet. *IEEE Access 6* (2018), 13810–13824.

[224] Kovács, L., and Szabó, G. Conceptualization with incremental bron-kerbosch algorithm in big data architecture. *Acta Polytechnica Hungarica 13*, 2 (2016), 139–158.

[225] Kraska, T., Beutel, A., Chi, E. H., Dean, J., and Polyzotis, N. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data* (2018), ACM, pp. 489–504.

[226] Kulis, B., and Grauman, K. Kernelized locality-sensitive hashing for scalable image search. In *ICCV* (2009), vol. 9, pp. 2130–2137.

[227] Kulkarni, P., Ganesan, D., Shenoy, P., and Lu, Q. Senseye: a multi-tier camera sensor network. In *Proceedings of the 13th annual ACM international conference on Multimedia* (2005), pp. 229–238.

[228] Kumar, N. M., and Mallick, P. K. The internet of things: Insights into the building blocks, component interactions, and architecture layers. *Procedia computer science 132* (2018), 109–117.

[229] Kumar, V., and Chadha, A. Mining association rules in student's assessment data. *International Journal of Computer Science Issues (IJCSI) 9*, 5 (2012), 211.

[230] Lai, H., Pan, Y., Liu, Y., and Yan, S. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3270–3278.

[231] Lakshmanaprabu, S., Mohanty, S. N., Shankar, K., Arunkumar, N., and Ramirez, G. Optimal deep learning model for classification of lung cancer on ct images. *Future Generation Computer Systems 92* (2019), 374–382.

[232] Leeser, M., Miller, S., and Yu, H. Smart camera based on reconfigurable hardware enables diverse real-time applications. In *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (2004), IEEE, pp. 147–155.

[233] Lehman, T. J., and Carey, M. J. A study of index structures for main memory database management systems. Tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 1985.

[234] Lejsek, H., et al. Nv-tree: a scalable disk-based high-dimensional index.

[235] Lemire, D., Kaser, O., and Aouiche, K. Sorting improves word-aligned bitmap indexes. *Data & Knowledge Engineering 69*, 1 (2010), 3–28.

[236] Li, A., Deng, Z., Liu, X., and Zhao, Z. A cooperative camera surveillance method based on the principle of coarse-fine coupling boresight adjustment. *Precision Engineering 66* (2020), 99–109.

[237] Li, C., Chen, Z., Zheng, W., Wu, Y., and Cao, J. Bah: A bitmap index compression algorithm for fast data retrieval. In *2016 IEEE 41st Conference on Local Computer Networks (LCN)* (2016), IEEE, pp. 697–705.

[238] Li, P., Zhu, X., Zhang, X., Ren, P., and Wang, L. Hash code reconstruction for fast similarity search. *IEEE Signal Processing Letters 26*, 5 (2019), 695–699.

[239] Li, S., Da Xu, L., and Zhao, S. The internet of things: a survey. *Information Systems Frontiers 17*, 2 (2015), 243–259.

[240] Li, W. Camera sensor activation scheme for target tracking in wireless visual sensor networks. *International Journal of Distributed Sensor Networks 9*, 4 (2013), 397537.

[241] Li, W.-J., Wang, S., and Kang, W.-C. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855* (2015).

[242] Li, Z., Zhang, X., Müller, H., and Zhang, S. Large-scale retrieval for medical image analytics: A comprehensive review. *Medical image analysis 43* (2018), 66–84.

[243] Lin, C. H., Lv, T., Wolf, W., and Ozer, I. B. A peer-to-peer architecture for distributed real-time gesture recognition. In *2004 IEEE International Conference on Multimedia and Expo (ICME)(IEEE Cat. No. 04TH8763)* (2004), vol. 1, IEEE, pp. 57–60.

[244] Lin, G., Shen, C., Shi, Q., Van den Hengel, A., and Suter, D. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1963–1970.

[245] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., and Zhao, W. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal 4*, 5 (2017), 1125–1142.

[246] Lin, Y., Zheng, Z., Zhang, H., Gao, C., and Yang, Y. Bayesian query expansion for multi-camera person re-identification. *Pattern Recognition Letters 130* (2020), 284–292.

[247] Liu, D., Guan, X., Du, Y., and Zhao, Q. Measuring indoor occupancy in intelligent buildings using the fusion of vision sensors. *Measurement Science and Technology 24*, 7 (2013), 074023.

[248] Liu, J., Tong, X., Li, W., Wang, T., Zhang, Y., and Wang, H. Automatic player detection, labeling and tracking in broadcast soccer video. *Pattern Recognition Letters 30*, 2 (2009), 103–113.

[249] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165* (2018).

[250] Liu, L., Zhang, X., and Ma, H. Optimal node selection for target localization in wireless camera sensor networks. *IEEE Transactions on Vehicular Technology 59*, 7 (2010), 3562–3576.

[251] Liu, T., Moore, A. W., and Gray, A. New algorithms for efficient high-dimensional nonparametric classification. *Journal of Machine Learning Research 7*, Jun (2006), 1135–1158.

[252] Liu, W., Mu, C., Kumar, S., and Chang, S.-F. Discrete graph hashing. In *Advances in neural information processing systems* (2014), pp. 3419–3427.

[253] Liu, W., Wang, J., Ji, R., Jiang, Y.-G., and Chang, S.-F. Supervised hashing with kernels. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), IEEE, pp. 2074–2081.

[254] Liu, X. A survey on clustering routing protocols in wireless sensor networks. *sensors 12*, 8 (2012), 11113–11153.

[255] Liu, X., Mu, Y., Zhang, D., Lang, B., and Li, X. Large-scale unsupervised hashing with shared structure learning. *IEEE transactions on cybernetics 45*, 9 (2014), 1811–1822.

[256] Liu, Y., Kong, L., Chen, G., Xu, F., and Wang, Z. Light-weight ai and iot collaboration for surveillance video pre-processing. *Journal of Systems Architecture* (2020), 101934.

[257] Lu, X., Zhu, L., Cheng, Z., Li, J., Nie, X., and Zhang, H. Flexible online multimodal hashing for large-scale multimedia retrieval. In *Proceedings of the 27th ACM International Conference on Multimedia* (2019), ACM, pp. 1129–1137.

[258] Lu, Y., and Payandeh, S. Cooperative hybrid multi-camera tracking for people surveillance. *Canadian Journal of Electrical and Computer Engineering 33*, 3/4 (2008), 145–152.

[259] Luo, X., Wang, F., and Luo, M. Collaborative target tracking in lopor with multi-camera. *Optik 127*, 23 (2016), 11588–11598.

[260] Lv, Q., Josephson, W., Wang, Z., Charikar, M., and Li, K. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases* (2007), VLDB Endowment, pp. 950–961.

[261] Lyon, D. *Surveillance society: Monitoring everyday life*. McGraw-Hill Education (UK), 2001.

[262] Maaloul, B., Taleb-Ahmed, A., Niar, S., Harb, N., and Valderrama, C. Adaptive video-based algorithm for accident detection on highways. In *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)* (2017), IEEE, pp. 1–6.

[263] MacNicol, R., and French, B. Sybase iq multiplex-designed for analytics. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30* (2004), VLDB Endowment, pp. 1227–1230.

[264] Mahmood, A. R., Punni, S., and Aref, W. G. Spatio-temporal access methods: a survey (2010-2017). *GeoInformatica 23*, 1 (2019), 1–36.

[265] Mahmood, Z. *Fog computing: concepts, frameworks and technologies*. Springer, 2018.

[266] Mahmud, R., Kotagiri, R., and Buyya, R. Fog computing: A taxonomy, survey and future directions. In *Internet of everything*. Springer, 2018, pp. 103–130.

[267] Mahoney, J., and LeHong, H. Innovation insight: the 'internet of everything'innovation will transform business. *Gartner Online, January 3* (2012).

[268] Malhotra, S., Doja, M. N., Alam, B., and Alam, M. Skipnet-octree based indexing technique for cloud database management system. *International Journal of Information Technology and Web Engineering (IJITWE) 13*, 3 (2018), 1–13.

[269] Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A. N., and Theodoridis, Y. *R-trees: Theory and Applications*. Springer Science & Business Media, 2010.

[270] Mao, R., Xu, H., Wu, W., Li, J., Li, Y., and Lu, M. Overcoming the challenge of variety: big data abstraction, the next evolution of data management for aal communication systems. *IEEE Communications Magazine 53*, 1 (2015), 42–47.

[271] Mao, R., Zhang, P., Li, X., Liu, X., and Lu, M. Pivot selection for metric-space indexing. *International Journal of Machine Learning and Cybernetics 7*, 2 (2016), 311–323.

[272] Marah, R., and El Hibaoui, A. Algorithms for smart grid management. *Sustainable cities and society 38* (2018), 627–635.

[273] Markakis, E., Mastorakis, G., Mavromoustakis, C. X., and Pallis, E. *Cloud and fog computing in 5G mobile networks: emerging advances and applications*. Institution of Engineering & Technology, 2017.

[274] Martinel, N. On a distributed video surveillance system to track persons in camera networks. *ELCVIA Electronic Letters on Computer Vision and Image Analysis 14*, 3 (2015), 39–41.

[275] Matsuyama, T., and Ukita, N. Real-time multitarget tracking by a cooperative distributed vision system. *Proceedings of the IEEE 90*, 7 (2002), 1136–1150.

[276] May, Z. B. Real-time alert system for home surveillance. In *2012 IEEE International Conference on Control System, Computing and Engineering* (2012), IEEE, pp. 501–505.

[277] McNames, J. A nearest trajectory strategy for time series prediction. In *Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling* (1998), KU Leuven Belgium, pp. 112–128.

[278] Mekonnen, T., Harjula, E., Heikkinen, A., Koskela, T., and Ylianttila, M. Energy efficient event driven video streaming surveillance using sleepycam. In *2017 IEEE International Conference on Computer and Information Technology (CIT)* (2017), IEEE, pp. 107–113.

[279] Mekonnen, T., Harjula, E., Koskela, T., and Ylianttila, M. sleepycam: power management mechanism for wireless video-surveillance cameras. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)* (2017), IEEE, pp. 91–96.

[280] Mell, P., and Grance, T. The nist definition of cloud computing (draft), nist spec. *Publ 800* (2011), 145.

[281] Mell, P., Grance, T., et al. The nist definition of cloud computing.

[282] Mendel, J. M., and Korjani, M. M. On establishing nonlinear combinations of variables from small to big data for use in later processing. *Information Sciences 280* (2014), 98–110.

[283] Merkwirth, C., Parlitz, U., and Lauterborn, W. Fast nearest-neighbor searching for nonlinear signal processing. *Physical Review E 62*, 2 (2000), 2089.

[284] Micheloni, C., Foresti, G. L., and Snidaro, L. A cooperative multicamera system for video-surveillance of parking lots. 5–5(1).

[285] Micó, M. L., Oncina, J., and Vidal, E. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing time and memory requirements. *Pattern Recognition Letters 15*, 1 (1994), 9–17.

[286] Mishra, S., and Chaurasiya, S. K. Cluster based coverage enhancement for directional sensor networks. In *Next Generation Computing Technologies (NGCT), 2015 1st International Conference on* (2015), IEEE, pp. 212–216.

[287] Monahan, T. Counter-surveillance as political intervention? *Social semiotics 16*, 4 (2006), 515–534.

[288] Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R. H., Morrow, M. J., and Polakos, P. A. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials 20*, 1 (2017), 416–464.

[289] Mourão, A., and Magalhães, J. Towards cloud distributed image indexing by sparse hashing. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval* (2019), ACM, pp. 288–296.

[290] Mu, Y., Shen, J., and Yan, S. Weakly-supervised hashing in kernel space. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), IEEE, pp. 3344–3351.

[291] Mu, Y., and Yan, S. Non-metric locality-sensitive hashing. In *Twenty-Fourth AAAI Conference on Artificial Intelligence* (2010).

[292] Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M. A., Choudhury, N., and Kumar, V. Security and privacy in fog computing: Challenges. *IEEE Access 5* (2017), 19293–19304.

[293] Murphey, R., and Pardalos, P. M. *Cooperative control and optimization*, vol. 66. Springer Science & Business Media, 2002.

[294] Nagarkar, P., Candan, K. S., and Bhat, A. Compressed spatial hierarchical bitmap (cshb) indexes for efficiently processing spatial range query workloads. *Proceedings of the VLDB Endowment 8*, 12 (2015), 1382–1393.

[295] Naha, R. K., Garg, S., Georgakopoulos, D., Jayaraman, P. P., Gao, L., Xiang, Y., and Ranjan, R. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE access 6* (2018), 47980–48009.

[296] Nalavade, J. E., and Murugan, T. S. Hrneuro-fuzzy: Adapting neuro-fuzzy classifier for recurring concept drift of evolving data streams using rough set theory and holoentropy. *Journal of King Saud University-Computer and Information Sciences 30*, 4 (2018), 498–509.

[297] Nashipudimath, M. M., and Shinde, S. K. Indexing in big data. In *Computing, Communication and Signal Processing*. Springer, 2019, pp. 133–142.

[298] Natarajan, P., Atrey, P. K., and Kankanhalli, M. Multi-camera coordination and control in surveillance systems: A survey. *ACM Trans. Multimedia Comput. Commun. Appl. 11*, 4 (2015), 57.

[299] Natesha, B., and Guddeti, R. M. R. Fog-based video surveillance system for smart city applications. In *Evolution in Computational Intelligence*. Springer, 2021, pp. 747–754.

[300] Navarro, G., and Reyes, N. New dynamic metric indices for secondary memory. *Information Systems 59* (2016), 48–78.

[301] Nebili, W., Farou, B., and Seridi, H. Using resources competition and memory cell development to select the best gmm for background subtraction. *International Journal of Strategic Information Technology and Applications (IJSITA) 10*, 2 (2019), 21–43.

[302] Nebili, W., Farou, B., and Seridi, H. Background subtraction using artificial immune recognition system and single gaussian (airs-sg). *Multimedia Tools and Applications 79*, 35 (2020), 26099–26121.

[303] Newell, A., and Akkaya, K. Distributed collaborative camera actuation for redundant data elimination in wireless multimedia sensor networks. *Ad Hoc Networks 9*, 4 (2011), 514–527.

[304] Nguyen, N. T., Venkatesh, S., West, G., and Bui, H. H. Multiple camera coordination in a surveillance system. *ACTA Automatica Sinica 29*, 3 (2003), 408–422.

[305] Ni, J., Zhang, K., Lin, X., and Shen, X. S. Securing fog computing for internet of things applications: Challenges and solutions. *IEEE Communications Surveys & Tutorials 20*, 1 (2017), 601–628.

[306] Ni, X.-w., and Xie, J. Bank video surveillance system design and analysis. *Computer Knowledge and Technology*, 5 (2013), 23.

[307] Nicheporchuk, V., Gryazin, I., and Favorskaya, M. N. Framework for intelligent wildlife monitoring. In *International Conference on Intelligent Decision Technologies* (2020), Springer, pp. 167–177.

[308] Nieto, M. *Public video surveillance: is it an effective crime prevention tool?* California Research Bureau, California State Library Sacramento, 1997.

[309] Nilsson, F., et al. *Intelligent network video: Understanding modern video surveillance systems*. CRC Press, 2016.

[310] Ning, H., Farha, F., Mohammad, Z. N., and Daneshmand, M. A survey and tutorial on "connection exploding meets efficient communication" in the internet of things. *IEEE Internet of Things Journal 7*, 11 (2020), 10733–10744.

[311] Norouzi, M., and Blei, D. M. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (2011), Citeseer, pp. 353–360.

[312] Obraczka, K., Manduchi, R., and Garcia-Luna-Aveces, J. Managing the information flow in visual sensor networks. In *The 5th International Symposium on Wireless Personal Multimedia Communications* (2002), vol. 3, IEEE, pp. 1177–1181.

[313] Olagoke, A. S., Ibrahim, H., and Teoh, S. S. Literature survey on multi-camera system and its application. *IEEE Access 8* (2020), 172892–172922.

[314] Oliveira, P. H., Traina Jr, C., and Kaster, D. S. Clap, acir and scoop: Novel techniques for improving the performance of dynamic metric access methods. *Information Systems 72* (2017), 117–135.

[315] Omohundro, S. M. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[316] O'Neil, P., and Quass, D. Improved query performance with variant indexes. *ACM Sigmod Record 26*, 2 (1997), 38–49.

[317] Ooi, B. C. Spatial kd-tree: A data structure for geographic database. In *Datenbanksysteme in Büro, Technik und Wissenschaft* (1987), Springer, pp. 247–258.

[318] Otair, D., et al. Approximate k-nearest neighbour based spatial clustering using kd tree. *arXiv preprint arXiv:1303.1951* (2013).

[319] Pan, B., Zheng, Y., Wilkie, D., and Shahabi, C. Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems* (2013), pp. 344–353.

[320] Pan, Y., Thulasiraman, P., and Wang, Y. Overview of cloudlet, fog computing, edge computing, and dew computing. In *Proceedings of The 3rd International Workshop on Dew Computing* (2018), pp. 20–23.

[321] Paredes, R. U., and Navarro, G. Egnat: A fully dynamic metric access method for secondary memory. In *2009 Second International Workshop on Similarity Search and Applications* (Aug 2009), pp. 57–64.

[322] Patel, F. S., and Kasat, D. Hashing based indexing techniques for content based image retrieval: A survey. In *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)* (2017), IEEE, pp. 279–283.

[323] Pavlidis, I., and Morellas, V. Two examples of indoor and outdoor surveillance systems: motivation, design, and testing. In *Video-Based Surveillance Systems*. Springer, 2002, pp. 39–50.

[324] Perdacher, M., Plant, C., and Böhm, C. Cache-oblivious high-performance similarity join. In *Proceedings of the 2019 International Conference on Management of Data* (2019), ACM, pp. 87–104.

[325] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials 16*, 1 (2013), 414–454.

[326] Petković, M., Vujović, I., and Kuzmanić, I. An overview on horizon detection methods in maritime video surveillance. *Transactions on Maritime Science 9*, 01 (2020), 106–112.

[327] Pillai, K. G., Sturlaugson, L., Banda, J. M., and Angryk, R. A. Extending high-dimensional indexing techniques pyramid and iminmax (θ): lessons learned. In *British National Conference on Databases* (2013), Springer, pp. 253–267.

[328] Pola, I. R., Traina, A. J., Traina, C., and Kaster, D. S. Improving metric access methods with bucket files. In *International Conference on Similarity Search and Applications* (2015), Springer, pp. 65–76.

[329] Pola, I. R. V., Traina, Jr, C., and Traina, A. J. M. The mm-tree: A memory-based metric tree without overlap between nodes. *ADBIS 2007 LNCS 4690* (2007), 157–171.

[330] Pola, I. R. V., Traina Jr, C., and Traina, A. J. M. The nobh-tree: Improving in-memory metric access methods by using metric hyperplanes with non-overlapping nodes. *Data & Knowledge Engineering 94* (2014), 65–88.

[331] Pourghebleh, B., and Navimipour, N. J. Data aggregation mechanisms in the internet of things: A systematic review of the literature and recommendations for future research. *Journal of Network and Computer Applications 97* (2017), 23–34.

[332] Pramanik, P. K. D., Mukherjee, B., Pal, S., Pal, T., and Singh, S. P. Green smart building: Requisites, architecture, challenges, and use cases. In *Green Building Management and Smart Automation*. IGI Global, 2020, pp. 1–50.

[333] Prati, A., Shan, C., and Wang, K. I.-K. Sensors, vision and networks: From video surveillance to activity recognition and health monitoring. *Journal of Ambient Intelligence and Smart Environments 11*, 1 (2019), 5–22.

[334] Prati, A., Vezzani, R., Benini, L., Farella, E., and Zappi, P. An integrated multi-modal sensor network for video surveillance. In *Proceedings of the third ACM international workshop on Video surveillance & sensor networks* (2005), pp. 95–102.

[335] Priyadarshini, S. B. B., and Panigrahi, S. Centralised cum sub-centralised scheme for multi-event coverage and optimum camera activation in wireless multimedia sensor networks. *IET Networks 4*, 6 (2015), 314–328.

[336] Priyadarshini, S. B. B., and Panigrahi, S. Redundant data minimization using minimal mean distant scalar leader selection for event driven camera actuation. In *2016 International Conference on Information Technology (ICIT)* (2016), IEEE, pp. 142–147.

[337] Priyadarshini, S. B. B., and Panigrahi, S. A distributed approach based on hierarchical scalar leader selection for enhanced event coverage in wireless multimedia sensor networks. In *International Conference on Distributed Computing and Internet Technology* (2017), Springer, pp. 3–14.

[338] Qiu, T., Chi, J., Zhou, X., Ning, Z., Atiquzzaman, M., and Wu, D. O. Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Communications Surveys Tutorials* (2020), 1–1.

[339] Rabinovich, M., Xiao, Z., and Aggarwal, A. Computing on the edge: A platform for replicating internet applications. In *Web content caching and distribution*. Springer, 2004, pp. 57–77.

[340] Rachkovskij, D. Distance-based index structures for fast similarity search. *Cybernetics and Systems Analysis 53*, 4 (2017), 636–658.

[341] Rahimi, M., Baer, R., Iroezi, O. I., Garcia, J. C., Warrior, J., Estrin, D., and Srivastava, M. Cyclops: in situ image sensing and interpretation in wireless sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems* (2005), pp. 192–204.

[342] Rahman, H., Ahmed, N., and Hussain, I. Comparison of data aggregation techniques in internet of things (iot). In *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)* (2016), IEEE, pp. 1296–1300.

[343] Rahmani, A. M., Liljeberg, P., Preden, J.-S., and Jantsch, A. *Fog computing in the internet of things: Intelligence at the edge*. Springer, 2017.

[344] Rai, M., Husain, A. A., Maity, T., and Yadav, R. K. Advance intelligent video surveillance system (aivss): a future aspect. In *Intelligent Video Surveillance*. IntechOpen, 2018.

[345] Rani, S., Ahmed, S. H., Talwar, R., Malhotra, J., and Song, H. Iomt: A reliable cross layer protocol for internet of multimedia things. *IEEE Internet of things Journal 4*, 3 (2017), 832–839.

[346] Razente, H., and Barioni, M. C. N. Storing data once in m-tree and pm-tree. In *International Conference on Similarity Search and Applications* (2019), Springer, pp. 18–31.

[347] Reddy, G. T., Kaluri, R., Reddy, P. K., Lakshmanna, K., Koppu, S., and Rajput, D. S. A novel approach for home surveillance system using iot adaptive security. In *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India* (2019).

[348] Rekha, G., Tyagi, A. K., and Anuradha, N. Integration of fog computing and internet of things: An useful overview. In *Proceedings of ICRIC 2019*. Springer, 2020, pp. 91–102.

[349] Remagnino, P., Shihab, A. I., and Jones, G. A. Distributed intelligence for multi-camera visual surveillance. *Pattern recognition 37*, 4 (2004), 675–689.

[350] Rinner, B., and Wolf, W. An introduction to distributed smart cameras. *Proceedings of the IEEE 96*, 10 (2008), 1565–1575.

[351] Robert, H. Z. Hobbes' internet timeline 25, 2018.

[352] Robinson, J. T. The kdb-tree: a search structure for large multidimensional dynamic indexes. In *Proceedings of the 1981 ACM SIGMOD international conference on Management of data* (1981), ACM, pp. 10–18.

[353] Roman, S., Axler, S., and Gehring, F. *Advanced linear algebra*, vol. 3. Springer, 2005.

[354] Rosas-Arias, L., Portillo-Portillo, J., Hernandez-Suarez, A., Olivares-Mercado, J., Sanchez-Perez, G., Toscano-Medina, K., Perez-Meana, H., Sandoval Orozco, A. L., and García Villalba, L. J. Vehicle counting in video sequences: an incremental subspace learning approach. *Sensors 19*, 13 (2019), 2848.

[355] Ross, B. What's the harm? the ethics of intelligence collection. *Intelligence and National Security 27*, 1 (2012), 93–117.

[356] Sakovich, N. What is the internet of everything (ioe)?, 2019.

[357] Salim, A., and Ramdan, H. An efficient distributed collaborative camera actuation algorithm for redundant data elimination for event detection and monitoring in wireless multimedia sensor networks. *International Journal of Computer Applications 975* (2016), 8887.

[358] Samet, H. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR) 16*, 2 (1984), 187–260.

[359] Samson, G., Joan, L., Usman, M. M., Showole, A. A., and Hadeel, H. J. Large spatial database indexing with ax-tree. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology 3*, 3 (2018), 759–773.

[360] SanMiguel, J. C., and Cavallaro, A. Cost-aware coalitions for collaborative tracking in resource-constrained camera networks. *IEEE Sensors Journal 15*, 5 (2014), 2657–2668.

[361] Santamaria, A. F., Raimondo, P., Palmieri, N., Tropea, M., and De Rango, F. Cooperative video-surveillance framework in internet of things (iot) domain. In *The Internet of Things for Smart Urban Ecosystems*. Springer, 2019, pp. 305–331.

[362] Santos, M. G. d., Ameyed, D., Petrillo, F., Jaafar, F., and Cheriet, M. Internet of things architectures: A comparative study. *arXiv preprint arXiv:2004.12936* (2020).

[363] Santucci, G. The internet of things: Between the revolution of the internet and the metamorphosis of objects. *Vision and Challenges for Realising the Internet of Things* (2010), 11–24.

[364] Saqlain, M., Piao, M., Shim, Y., and Lee, J. Y. Framework of an iot-based industrial data management for smart manufacturing. *Journal of Sensor and Actuator Networks 8*, 2 (2019), 25.

[365] Sarkar, S., Chatterjee, S., and Misra, S. Assessment of the suitability of fog computing in the context of internet of things. *IEEE Transactions on Cloud Computing 6*, 1 (2015), 46–59.

[366] Satuluri, V., and Parthasarathy, S. Bayesian locality sensitive hashing for fast similarity search. *Proceedings of the VLDB Endowment 5*, 5 (2012), 430–441.

[367] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing 8*, 4 (2009), 14–23.

[368] Schaeffer, S. E. Graph clustering. *Computer science review 1*, 1 (2007), 27–64.

[369] Scott, S. J., Nguyen, T. T., Brandt, D. D., Gibart, T., and Dotson, G. D. Recognition-based industrial automation control with person and object discrimination, 2016. US Patent 9,393,695.

[370] Seema, A., and Reisslein, M. Towards efficient wireless video sensor networks: A survey of existing node architectures and proposal for a flexi-wvsnp design. *IEEE Communications Surveys & Tutorials 13*, 3 (2010), 462–486.

[371] Sellis, T., Roussopoulos, N., and Faloutsos, C. The r+-tree: A dynamic index for multi-dimensional objects. Tech. rep., 1987.

[372] Sethi, P., and Sarangi, S. R. Internet of things: architectures, protocols, and applications. *Journal of Electrical and Computer Engineering 2017* (2017).

[373] Shaikh, H. S., Jadhav, S. P., Chavan, S. S., Kaninde, G. M., and Sanap, M. Automatic signal scheduling for efficient traffic management. *International Journal For Research & Development in Technology 5*, 6 (2020), 40–45.

[374] Shakhnarovich, G. *Learning task-specific similarity*. PhD thesis, Massachusetts Institute of Technology, 2005.

[375] Shang, L., Yang, L., Wang, F., Chan, K.-P., and Hua, X.-S. Real-time large scale near-duplicate web video retrieval. In *Proceedings of the 18th ACM international conference on Multimedia* (2010), ACM, pp. 531–540.

[376] Sharmin, S., Nur, F. N., Razzaque, M. A., Rahman, M. M., Alelaiwi, A., Hassan, M. M., and Rahman, S. M. M. α-overlapping area coverage for clustered directional sensor networks. *Computer Communications 109* (2017), 89–103.

[377] Sheikh, Y., Javed, O., and Shah, M. Object association across multiple cameras., 2009.

[378] Shen, F., Shen, C., Liu, W., and Tao Shen, H. Supervised discrete hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 37–45.

[379] Shen, F., Shen, C., Shi, Q., Van Den Hengel, A., and Tang, Z. Inductive hashing on manifolds. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2013), pp. 1562–1569.

[380] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. Edge computing: Vision and challenges. *IEEE Internet of Things Journal 3*, 5 (2016), 637–646.

[381] Shi, W., and Dustdar, S. The promise of edge computing. *Computer 49*, 5 (2016), 78–81.

[382] Shi, X., Xing, F., Xu, K., Sapkota, M., and Yang, L. Asymmetric discrete graph hashing. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[383] Shoham, Y. Agent-oriented programming. *Artificial intelligence 60*, 1 (1993), 51–92.

[384] Shrivastava, A., and Li, P. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems* (2014), pp. 2321–2329.

[385] Siddiqa, A., Hashem, I. A. T., Yaqoob, I., Marjani, M., Shamshirband, S., Gani, A., and Nasaruddin, F. A survey of big data management: Taxonomy and state-of-the-art. *Journal of Network and Computer Applications 71* (2016), 151–166.

[386] Sikandar, T., Ghazali, K. H., and Rabbi, M. F. Atm crime detection using image processing integrated video surveillance: a systematic review. *Multimedia Systems 25*, 3 (2019), 229–251.

[387] Skala, K., Davidovic, D., Afgan, E., Sovic, I., and Sojat, Z. Scalable distributed computing hierarchy: Cloud, fog and dew computing. *Open Journal of Cloud Computing (OJCC) 2*, 1 (2015), 16–24.

[388] Skopal, T. Pivoting m-tree: A metric access method for efficient similarity search. In *DATESO* (2004), vol. 4, Citeseer, pp. 27–37.

[389] Skopal, T., Pokornỳ, J., Krátkỳ, M., and Snášel, V. Revisiting m-tree building principles. In *East European Conference on Advances in Databases and Information Systems* (2003), Springer, pp. 148–162.

[390] Skopal, T., Pokornỳ, J., and Snasel, V. Pm-tree: Pivoting metric tree for similarity search in multimedia databases. In *ADBIS (Local Proceedings)* (2004).

[391] Sommerlade, E., and Reid, I. Phd forum: Probabilistic surveillance with multiple active cameras. In *2009 Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)* (2009), IEEE, pp. 1–2.

[392] Soro, S., and Heinzelman, W. A survey of visual sensor networks. *Advances in multimedia 2009* (2009).

[393] SOURYA, B. History of cloud computing, 2011.

[394] Sprenger, S., Schäfer, P., and Leser, U. Bb-tree: A main-memory index structure for multidimensional range queries. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (2019), IEEE, pp. 1566–1569.

[395] Sprenger, S., Schäfer, P., and Leser, U. Bb-tree: A main-memory index structure for multidimensional range queries. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (2019), IEEE, pp. 1566–1569.

[396] Sprenger, S., Schäfer, P., and Leser, U. Bb-tree: A practical and efficient main-memory index structure for multidimensional workloads. In *EDBT* (2019), pp. 169–180.

[397] Sriharsha, C., Kumar, P., and Jindal, A. Upbit with parallelized merge. In *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (2019), IEEE, pp. 625–629.

[398] Srinivasan, V., and Carey, M. J. Performance of b-tree concurrency control algorithms. In *Proceedings of the 1991 ACM SIGMOD international conference on Management of data* (1991), pp. 416–425.

[399] Srinivasan, V., and Carey, M. J. Performance of b+ tree concurrency control algorithms. *The VLDB Journal 2*, 4 (1993), 361–406.

[400] Stankovic, J. A. Misconceptions about real-time computing: A serious problem for next-generation systems. *Computer 21*, 10 (1988), 10–19.

[401] Stipaničev, D., Vuko, T., Krstinić, D., Štula, M., and Bodrožić, L. Forest fire protection by advanced video detection system-croatian experiences. In *Third TIEMS Workshop-Improvement of Disaster Managment Systems-local and global trends* (2006).

[402] Stojmenovic, I., and Wen, S. The fog computing paradigm: Scenarios and security issues. In *2014 federated conference on computer science and information systems* (2014), IEEE, pp. 1–8.

[403] Strecha, C., Bronstein, A., Bronstein, M., and Fua, P. Ldahash: Improved matching with smaller descriptors. *IEEE transactions on pattern analysis and machine intelligence 34*, 1 (2011), 66–78.

[404] Surbhi, S. A deep dive into the internet of everything – the next big tech thing, 2013.

[405] Sycara, K. Multiagent systems. *AI magazine 19*, 2 (1998), 79–79.

[406] Tabassum, N., and Ahmed, T. A theoretical study on classifier ensemble methods and its applications. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (2016), IEEE, pp. 374–378.

[407] Tan, L., and Wang, N. Future internet: The internet of things. In *2010 3rd international conference on advanced computer theory and engineering (ICACTE)* (2010), vol. 5, IEEE, pp. 376–380.

[408] Tang, X., Han, B., and Chen, H. A hybrid index for multi-dimensional query in hbase. In *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)* (2016), IEEE, pp. 332–336.

[409] Tavli, B., Bicakci, K., Zilan, R., and Barcelo-Ordinas, J. M. A survey of visual sensor network platforms. *Multimedia Tools and Applications 60*, 3 (2012), 689–726.

[410] Tel, G. *Introduction to distributed algorithms*. Cambridge university press, 2000.

[411] Tessens, L., Morbee, M., Aghajan, H., and Philips, W. Camera selection for tracking in distributed smart camera networks. *ACM Transactions on Sensor Networks 10*, 2 (2014).

[412] Thingom, I. B. Internet of things: design of a new layered architecture and study of some existing issues. *IOSR Journal of Computer Engineering* (2015), 26–30.

[413] Thomsen, F. K. The concepts of surveillance and sousveillance: A critical analysis. *Social Science Information 58*, 4 (2019), 701–713.

[414] Tian, L., Fan, C., and Ming, Y. Learning spherical hashing based binary codes for face recognition. *Multimedia Tools and Applications 76*, 11 (2017), 13271–13299.

[415] Tian, Y.-l., Brown, L., Hampapur, A., Lu, M., Senior, A., and Shu, C.-f. Ibm smart surveillance system (s3): event based video surveillance system with an open and extensible framework. *Machine Vision and Applications 19*, 5-6 (2008), 315–327.

[416] Tichelaar, S. *A coordination component framework for open distributed systems*. PhD thesis, Faculty of Science and Engineering, 1997.

[417] Tinka, A., Rafiee, M., and Bayen, A. M. Floating sensor networks for river studies. *IEEE Systems Journal 7*, 1 (2012), 36–49.

[418] Tomita, E., Tanaka, A., and Takahashi, H. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical computer science 363*, 1 (2006), 28–42.

[419] Traina, C., Traina, A., Faloutsos, C., and Seeger, B. Fast indexing and visualization of metric data sets using slim-trees. *IEEE Transactions on Knowledge and Data Engineering 14*, 2 (2002), 244–260.

[420] Traina, C., Traina, A., Seeger, B., and Faloutsos, C. Slim-trees: High performance metric trees minimizing overlap between nodes. In *International Conference on Extending Database Technology* (2000), Springer, pp. 51–65.

[421] Tsakanikas, V., and Dagiuklas, T. Video surveillance systems-current status and future trends. *Computers & Electrical Engineering 70* (2018), 736–753.

[422] Tseng, B. L., Lin, C.-Y., and Smith, J. R. Real-time video surveillance for traffic monitoring using virtual line analysis. In *Proceedings. IEEE International Conference on Multimedia and Expo* (2002), vol. 2, IEEE, pp. 541–544.

[423] Uehara, M. Mist computing: Linking cloudlet to fogs. In *International Conference on Computational Science/Intelligence & Applied Informatics* (2017), Springer, pp. 201–213.

[424] Uhlmann, J. K. Satisfying general proximity/similarity queries with metric trees. *Information processing letters 40*, 4 (1991), 175–179.

[425] Ukita, N., and Matsuyama, T. Real-time multi-target tracking by cooperative distributed active vision agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2* (2002), pp. 829–838.

[426] Valera, M., and Velastin, S. A. Intelligent distributed surveillance systems: a review. *IEE Proceedings-Vision, Image and Signal Processing 152*, 2 (2005), 192–204.

[427] Van Steen, M., and Tanenbaum, A. S. A brief introduction to distributed systems. *Computing 98*, 10 (2016), 967–1009.

[428] Vancea, B. A. Cluster-computing and parallelization for the multi-dimensional ph-index. Master's thesis, ETH Zurich, 2015.

[429] Vaquero, L. M., and Rodero-Merino, L. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review 44*, 5 (2014), 27–32.

[430] Verissimo, P., and Rodrigues, L. *Distributed systems for system architects*, vol. 1. Springer Science & Business Media, 2012.

[431] Verley, J.-L. Espaces métriques. *Dictionnaire des mathématiques; algèbre, analyse, géométrie, éd. Albin Michel, Paris* (2002), 652–653.

[432] Villalpando, L. E. B., April, A., and Abran, A. Dipar: a framework for implementing big data science in organizations. In *Continued Rise of the Cloud*. Springer, 2014, pp. 177–194.

[433] Visheratin, A. A., Mukhina, K. D., Visheratina, A. K., Nasonov, D., and Boukhanovsky, A. V. Multiscale event detection using convolutional quadtrees and adaptive geogrids. In *Proceedings of the 2nd ACM SIGSPATIAL Workshop on Analytics for Local Events and News* (2018), ACM, p. 1.

[434] Wan Yuchai, Liu Xiabi, W. Y. Cd-tree: A clustering-based dynamic indexing and retrieval approach. *Intelligent Data Analysis 21* (2017), 243–261.

[435] Wang, J., Kumar, S., and Chang, S.-F. Sequential projection learning for hashing with compact codes.

[436] Wang, J., Kumar, S., and Chang, S.-F. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 12 (2012), 2393–2406.

[437] Wang, J., Liu, W., Kumar, S., and Chang, S.-F. Learning to hash for indexing big data—a survey. *Proceedings of the IEEE 104*, 1 (2015), 34–57.

[438] Wang, J., Pan, J., and Esposito, F. Elastic urban video surveillance system using edge computing. In *Proceedings of the Workshop on Smart Internet of Things* (2017), Association for Computing Machinery, p. 7.

[439] Wang, J., Shen, H. T., Song, J., and Ji, J. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927* (2014).

[440] Wang, S., Li, C., and Shen, H. Equivalent continuous formulation of general hashing problem. *IEEE Transactions on Cybernetics* (2019), 1–11.

[441] Wang, X. Intelligent multi-camera video surveillance: A review. *Pattern recognition letters 34*, 1 (2013), 3–19.

[442] Wang, X., Meng, W., and Zhang, M. A novel information retrieval method based on r-tree index for smart hospital information system. *International Journal of Advanced Computer Research 9*, 42 (2019), 133–145.

[443] Wang, Y. Cloud-dew architecture. *International Journal of Cloud Computing 4*, 3 (2015), 199–210.

[444] Wang, Y. The initial definition of dew computing, 2015.

[445] Wang, Y. Definition and categorization of dew computing. *Open Journal of Cloud Computing (OJCC) 3*, 1 (2016), 1–7.

[446] Wang, Y., Lin, Y., and Yang, J. Kd-tree based clustering algorithm for fast face recognition on large-scale data. In *Seventh International Conference on Digital Image Processing (ICDIP 2015)* (2015), vol. 9631, International Society for Optics and Photonics, p. 96311I.

[447] Wang, Y., and Pan, Y. Cloud-dew architecture : realizing the potential of distributed database systems in unreliable networks.

[448] Wang, Y., Wang, D., and Fang, W. Automatic node selection and target tracking in wireless camera sensor networks. *Computers & Electrical Engineering 40*, 2 (2014), 484–493.

[449] Wang, Y., Yun, X., Wang, X., Wang, S., and Wu, Y. Lbfm: Multi-dimensional membership index for block-level data skipping. In *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)* (2017), IEEE, pp. 343–351.

[450] Wang, Y., Zhao, C., Wang, Z., Du, J., Liu, C., Yan, H., Wen, J., Hou, H., and Zhou, K. Mlb+-tree: A multi-level b+-tree index for multidimensional range query on seismic data. In *2018 5th International Conference on Systems and Informatics (ICSAI)* (2018), IEEE, pp. 1176–1181.

[451] Wang, Z., Luo, T., Xu, G., and Wang, X. A new indexing technique for supporting by-attribute membership query of multidimensional data. In *International Conference on Web-Age Information Management* (2013), Springer, pp. 266–277.

[452] Watve, A., Pramanik, S., Shahid, S., Meiners, C. R., and Liu, A. X. Topological transformation approaches to database query processing. *IEEE Transactions on Knowledge and Data Engineering 27*, 5 (2014), 1438–1451.

[453] Weber, R., and Blott, S. An approximation based data structure for similarity search. Tech. rep., Citeseer, 1997.

[454] Weber, R., Schek, H.-J., and Blott, S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB* (1998), vol. 98, pp. 194–205.

[455] Wei, X., Li, W.-X., Ran, C., Pi, C.-C., Ma, Y.-J., and Sheng, Y.-X. Architecture and scheduling method of cloud video surveillance system based on iot. In *International Conference on Algorithms and Architectures for Parallel Processing* (New York, NY, USA, 2015), Springer, pp. 551–560.

[456] Weihong, W., and Jiaoyang, T. Research on license plate recognition algorithms based on deep learning in complex environment. *IEEE Access 8* (2020), 91661–91675.

[457] Weiser, M. The computer for the 21st century. *ACM SIGMOBILE mobile computing and communications review 3*, 3 (1999), 3–11.

[458] Weiss, G. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.

[459] Weiss, Y., Torralba, A., and Fergus, R. Spectral hashing. In *Advances in neural information processing systems* (2009), pp. 1753–1760.

[460] Wen, Y., Chen, Z., Ma, G., Cao, J., Zheng, W., Peng, G., Li, S., and Huang, W.-L. Secompax: A bitmap index compression algorithm. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)* (2014), IEEE, pp. 1–7.

[461] White, D. A., and Jain, R. Similarity indexing with the ss-tree. In *Proceedings of the Twelfth International Conference on Data Engineering* (1996), IEEE, pp. 516–523.

[462] Won, M. Intelligent traffic monitoring systems for vehicle classification: A survey. *IEEE Access 8* (2020), 73340–73358.

[463] Wu, C., and Aghajan, H. Model-based human posture estimation for gesture analysis in an opportunistic fusion smart camera network. In *2007 IEEE Conference on Advanced Video and Signal Based Surveillance* (2007), IEEE, pp. 453–458.

[464] Wu, C., Zhu, J., Cai, D., Chen, C., and Bu, J. Semi-supervised nonlinear hashing using bootstrap sequential projection learning. *IEEE Transactions on Knowledge and Data Engineering 25*, 6 (2012), 1380–1393.

[465] Wu, J., Zhang, Y., Wang, J., Lin, C., Fu, Y., and Xing, C. Improving distributed similarity join in metric space with error-bounded sampling. *arXiv preprint arXiv:1905.05981* (2019).

[466] Wu, K., Otoo, E. J., and Shoshani, A. Compressing bitmap indexes for faster search operations. In *Proceedings 14th International Conference on Scientific and Statistical Database Management* (2002), IEEE, pp. 99–108.

[467] Wu, K., Otoo, E. J., and Shoshani, A. Optimizing bitmap indices with efficient compression. *ACM Transactions on Database Systems (TODS) 31*, 1 (2006), 1–38.

[468] Wu, M., Lu, T.-J., Ling, F.-Y., Sun, J., and Du, H.-Y. Research on the architecture of internet of things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)* (2010), vol. 5, IEEE, pp. V5–484.

[469] Wu, Y., Chen, Z., Wen, Y., Zheng, W., and Cao, J. Combat: A new bitmap index coding algorithm for big data. *Tsinghua Science and Technology 21*, 2 (2016), 136–145.

[470] Xia, R., Pan, Y., Lai, H., Liu, C., and Yan, S. Supervised hashing for image retrieval via image representation learning. In *Twenty-eighth AAAI conference on artificial intelligence* (2014).

[471] Xiang, C., and Li, X. General analysis on architecture and key technologies about internet of things. In *2012 IEEE International Conference on Computer Science and Automation Engineering* (2012), IEEE, pp. 325–328.

[472] Xie, F., and Ye, X. H. Endada: an efficient network design algorithm based on weighted graph for data aggregation in internet of things on marine ships. In *Applied Mechanics and Materials* (2015), vol. 740, Trans Tech Publ, pp. 648–651.

[473] Xie, L., Shen, J., Han, J., Zhu, L., and Shao, L. Dynamic multi-view hashing for online image retrieval. IJCAI.

[474] Xiong, L., and Xie, J. Research on construction of smart training room based on mobile cloud video surveillance technologies. In *International Conference on Computer Engineering and Networks* (2020), Springer, pp. 1113–1119.

[475] Xu, J., Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F., and Hao, H. Convolutional neural networks for text hashing. In *Twenty-Fourth International Joint Conference on Artificial Intelligence* (2015).

[476] Y.2060, R. I.-T. Overview of the internet of things, 2012.

[477] Yang, A. Y., Jafari, R., Sastry, S. S., and Bajcsy, R. Distributed recognition of human actions using wearable motion sensor networks. *Journal of Ambient Intelligence and Smart Environments 1*, 2 (2009), 103–115.

[478] Yang, K., Ding, X., Zhang, Y., Chen, L., Zheng, B., and Gao, Y. Distributed similarity queries in metric spaces. *Data Science and Engineering 4*, 2 (2019), 93–108.

[479] Yang, Y., Bai, P., Ge, N., Gao, Z., and Qiu, X. Lazy r-tree: The r-tree with lazy splitting algorithm. *Journal of Information Science* (2019), 0165551519828616.

[480] Yang, Y., Huang, J., Zhang, T., and Weinman, J. *Fog and Fogonomics: Challenges and Practices of Fog Computing, Communication, Networking, Strategy, and Economics.* John Wiley & Sons, 2020.

[481] Yang, Y., Shen, F., Shen, H. T., Li, H., and Li, X. Robust discrete spectral hashing for large-scale image semantic indexing. *IEEE Transactions on Big Data 1*, 4 (2015), 162–171.

[482] Yannuzzi, M., Milito, R., Serral-Gracià, R., Montero, D., and Nemirovsky, M. Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing. In *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)* (2014), IEEE, pp. 325–329.

[483] Yap, F., and Yen, H.-H. A survey on sensor coverage and visual data capturing/processing/transmission in wireless visual sensor networks. *Sensors 14*, 2 (2014), 3506–3527.

[484] Yeh, J.-S., Chang, C.-C., Chia, T.-L., Chiang, S.-Y., and Huang, P.-S. Cooperative dual camera surveillance system for real-time object searching and close-up viewing. In *2016 2nd International Conference on Intelligent Green Building and Smart Grid (IGBSG)* (2016), IEEE, pp. 1–5.

[485] Yi, S., Hao, Z., Qin, Z., and Li, Q. Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)* (2015), IEEE, pp. 73–78.

[486] Yianilos, P. N. Data structures and algorithms for nearest neighbor search in general metric spaces. *proceedings of the 4th Annual In ACM-SIAM Symposium on Discrete Algorithms* (1993), 311–321.

[487] Yianilos, P. N. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Soda* (1993), vol. 93, pp. 311–21.

[488] Yilmaz, A., Javed, O., and Shah, M. Object tracking: A survey. *Acm computing surveys (CSUR) 38*, 4 (2006), 13–es.

[489] Yoder, J., Medeiros, H., Park, J., and Kak, A. C. Cluster-based distributed face tracking in camera networks. *IEEE Transactions on Image Processing 19*, 10 (2010), 2551–2563.

[490] Yogish, H., Raju, G., and Manjunath, T. The descriptive study of knowledge discovery from web usage mining. *International Journal of Computer Science Issues (IJCSI) 8*, 5 (2011), 225.

[491] Yoon, C.-S., Jung, H.-S., Park, J.-W., Lee, H.-G., Yun, C.-H., and Lee, Y. W. A cloud-based utopia smart video surveillance system for smart cities. *Applied Sciences 10*, 18 (2020), 6572.

[492] Yu, D., and Zhang, A. Clustertree: Integration of cluster representation and nearest neighbor search for large datasets with high dimensionality. *IEEE Internati onal Conference on Multimedia and Expo, 2000 15* (2000), 1316–1337.

[493] Yun, M., and Yuxin, B. Research on the architecture and key technology of internet of things (iot) applied on smart grid. In *2010 International Conference on Advances in Energy Engineering* (2010), IEEE, pp. 69–72.

[494] Zamir, O., and Etzioni, O. Web document clustering: A feasibility demonstration. In *SIGIR* (1998), vol. 98, Citeseer, pp. 46–54.

[495] Zarifneshat, M., Khadivi, P., and Saidi, H. A semi-localized algorithm for cluster head selection for target tracking in grid wireless sensor networks. *Ad Hoc & Sensor Wireless Networks 25*, 3-4 (2015), 263–287.

[496] Zäschke, T. The ph-tree revisited. https://github.com/tzaeschke/phtree, 2015.

[497] Zäschke, T., Zimmerli, C., and Norrie, M. C. The ph-tree: a space-efficient storage structure and multi-dimensional index. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (2014), ACM, pp. 397–408.

[498] Zezula, P., Amato, G., Dohnal, V., and Batko, M. *Similarity search: the metric space approach*, vol. 32. Springer Science & Business Media, 2006.

[499] Zhang, C., and Zheng, W.-S. Semi-supervised multi-view discrete hashing for fast image search. *IEEE Transactions on Image Processing 26*, 6 (2017), 2604–2617.

[500] Zhang, J., Lei, Y., Chen, C., and Lin, F. Directional probability perceived nodes deployment based on particle swarm optimization. *International Journal of Distributed Sensor Networks 12*, 4 (2016), 2046392.

[501] Zhang, L., Zhang, Y., Gu, X., Tang, J., and Tian, Q. Scalable similarity search with topology preserving hashing. *IEEE Transactions on Image Processing 23*, 7 (2014), 3025–3039.

[502] Zhang, L., Zhang, Y., Tang, J., Gu, X., Li, J., and Tian, Q. Topology preserving hashing for similarity search. In *Proceedings of the 21st ACM international conference on Multimedia* (2013), ACM, pp. 123–132.

[503] Zhang, M., Sun, F., and Cheng, X. Architecture of internet of things and its key technology integration based-on rfid. In *2012 Fifth International Symposium on Computational Intelligence and Design* (2012), vol. 1, IEEE, pp. 294–297.

[504] Zhang, P., Zhou, C., Wang, P., Gao, B. J., Zhu, X., and Guo, L. E-tree: An efficient indexing structure for ensemble models on data streams. *IEEE Transactions on Knowledge and Data engineering 27*, 2 (2014), 461–474.

[505] Zhang, Q., Cheng, L., and Boutaba, R. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications 1*, 1 (2010), 7–18.

[506] Zhang, R., Lin, L., Zhang, R., Zuo, W., and Zhang, L. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Transactions on Image Processing 24*, 12 (2015), 4766–4779.

[507] Zhang, R., Ooi, B. C., and Tan, K.-L. Making the pyramid technique robust to query types and workloads. In *Proceedings. 20th International Conference on Data Engineering* (2004), IEEE, pp. 313–324.

[508] Zhang, R., Wu, L., Yang, Y., Wu, W., Chen, Y., and Xu, M. Multi-camera multi-player tracking with deep player identification in sports video. *Pattern Recognition 102* (2020), 107260.

[509] Zhang, S., Liu, X., Zhang, M., and Wo, T. Paindex: An online index system for vehicle trajectory data exploiting parallelism. In *2017 4th International Conference on Systems and Informatics (ICSAI)* (2017), IEEE, pp. 696–703.

[510] Zhao, S., and Yu, L. A sensor-service collaboration approach for target tracking in wireless camera networks. *China Communications 14*, 7 (2017), 1–13.

[511] Zheng, W., Liu, Y., Chen, Z., and Cao, J. Codis: A new compression scheme for bitmap indexes. In *Proceedings of the Symposium on Architectures for Networking and Communications Systems* (2017), IEEE Press, pp. 103–104.

[512] Zhou, X., Wang, G., Yu, J. X., and Yu, G. M+-tree: a new dynamical multi-dimensional index for metric spaces. In *Proceedings of the 14th Australasian database conference-Volume 17* (2003), Australian Computer Society, Inc., pp. 161–168.

[513] Zhou, X., Wang, G., Zhou, X., and Yu, G. Bm+-tree: A hyperplane-based index method for high-dimensional metric spaces. In *International Conference on Database Systems for Advanced Applications* (2005), Springer, pp. 398–409.

[514] Zhou, Y., Zhang, D., and Xiong, N. Post-cloud computing paradigms: a survey and comparison. *Tsinghua Science and Technology 22*, 6 (2017), 714–732.

[515] Zhu, H., Wang, J., Xie, K., and Ye, J. Detection of vehicle flow in video surveillance. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)* (2018), IEEE, pp. 528–532.

[516] Zhu, X., Liu, Z., and Yang, J. Model of collaborative uav swarm toward coordination and control mechanisms study. *Procedia Computer Science 51* (2015), 493–502.

[517] Zikria, Y. B., Afzal, M. K., and Kim, S. W. Internet of multimedia things (iomt): Opportunities, challenges and solutions. *Sensors 20*, 8 (2020), 2334.

AUTHOR'S PUBLICATION

❖ International publications

[**J3**] **Ala-Eddine Benrazek**, Kouahla Zinedine, Farou Brahim, Mohamed AmineFerrag, Seridi Hamid and Kurulay Muhammet, "An Efficient Indexing for IoT Massive Data based on Cloud-Fog Computing", WILEY Transactions on Emerging Telecommunications Technologies (ETT). DOI: 10.1002/ett.3868, 17 February, 2020.

[**J2**] **Ala-Eddine Benrazek**, Farou Brahim, Seridi Hamid, Kouahla Zinedine and Kurulay Muhammet, "Ascending Hierarchical Classification for Camera Clustering Based on FoV Overlaps for WMSN", IET Wireless Sensor Systems, 9.6, 382-388. DOI: 10.1049/iet-wss.2019.0030, 29 July, 2019.

[**J1**] **Ala-Eddine Benrazek**, Farou Brahim, and Kurulay Muhammet. "Efficient Camera Clustering Method Based on Overlapping FoVs for WMSNs." International Journal of Informatics and Applied Mathematics 1.1: 11-27, 2019.

❖ International Communications

[**C2**] Wafa Nebili, **Ala-Eddine Benrazek**, Muhammet Kurulay, Brahim Farou and Mohamed Amine Ferrag, "Enhancing the Field Coverage of UAV using Grey Wolf Optimizer", 1st International Conference on Innovative Trends in Computer Science (CITCS'2019) , Guelma, Algeria, 2019.

[**C1**] **Ala-Eddine Benrazek**, Brahim Farou, Hamid Seridi, "A Method of Grouping Cameras in a Video Surveillance System Network", 7th IEEE International Conference on Smart Communication in Network Technologies (SaCoNeT 2018), El-Ouad, Algeria, 2018.

❖ National Communications

[C2] **Ala-Eddine Benrazek**, Kouahla Zineddine, Farou Brahim and Seridi Hamid, "A Scalable Index for Internet-of-Things", 2nd Conference on Informatics and Applied Mathematics (IAM'2018), Guelma, Algeria, 2019.

[C1] **Ala-Eddine Benrazek**, Brahim Farou, Hamid Seridi, "A New Method of Grouping Cameras based on Overlapping Visual Fields in Wireless Multimedia Sensor Networks", 1st Conference on Informatics and Applied Mathematics (IAM'2018), Guelma, Algeria, 2018.