Multi-people tracking using a distributed camera network: application to a University Campus

Henrique Miguel Cardoso Matos

UMinho | 2023

Universidade do Minho
Escola de Engenharia

Henrique Miguel Cardoso Matos

Multi-people tracking using a distributed camera network: application to a University Campus

abril de 2023

Universidade do Minho
Escola de Engenharia

Henrique Miguel Cardoso Matos

Multi-people tracking using a distributed camera network: application to a University Campus

Dissertação de Mestrado em
Engenharia de Telecomunicações
e Informática

Trabalho efetuado sob a orientação do
Professor Doutor Henrique Santos

abril de 2023

Agradecimentos

A conclusão da minha dissertação de mestrado foi possível graças ao apoio imensurável e contribuições de várias pessoas que estiveram presentes sempre que precisei. Gostaria de aproveitar esta oportunidade para expressar a minha gratidão para todos eles.

Em primeiro lugar, gostaria de expressar a minha gratidão ao meu orientador, Professor Henrique Santos, pela sua orientação e apoio inestimáveis ao longo da minha pesquisa. A sua experiência e conhecimento na área foram fundamentais para moldar a direção desta dissertação. Ele não só forneceu o seu feedback sobre o meu trabalho, como também deu sugestões para métodos de pesquisa e deu acesso a recursos valiosos para o desenvolvimento. A sua orientação ajudou-me a desenvolver as minhas competências de pesquisa e a compreender melhores formas de ultrapassar os obstáculos que encontrei durante o processo de investigação.

Gostaria também de mostrar a minha apreciação pela oportunidade de desenvolver esta dissertação como parte do projeto de bolsa de investigação ”Lab4U&Spaces - Living Lab of Interactive Urban Space Solution” no Centro ALGORITMI, Universidade do Minho. Os recursos, instalações e apoio fornecidos pelo projeto foram fundamentais para a conclusão bem-sucedida da minha investigação. Agradeço à equipa do projeto e a todos os que contribuíram para ele.

Quero também expressar a minha gratidão aos meus amigos, que foram uma fonte incrível de apoio e encorajamento ao longo da minha jornada académica. Sou grato pelas inúmeras horas passadas a discutir ideias, a trabalhar em conjunto e a dar feedback. As contribuições deles foram inestimáveis e moldaram significativamente o curso da minha jornada académica. Através da presença deles, fui capaz de desenvolver competências e conhecimentos que impactaram significativamente o meu percurso académico.

Por último, quero expressar o meu profundo agradecimento à minha família por ser a minha fonte constante de apoio e inspiração ao longo da minha carreira educacional. O vosso amor e apoio deram-me força e determinação para perseguir os meus sonhos e superar os desafios que surgiram no meu percurso académico. Sem o vosso encorajamento e orientação, não estaria onde estou hoje. A vossa crença inabalável em mim foi uma fonte constante de motivação e estou eternamente grato pelo vosso apoio.

Acknowledgments

The completion of my master’s dissertation was made possible by the invaluable support and contributions of several individuals who were there for me whenever I needed them. I want to take this opportunity to express my gratitude to them.

Firstly, I would like to express my gratitude to my supervisor, Professor Henrique Santos, for his invaluable guidance and support throughout my research. His expertise and knowledge in the field have been instrumental in shaping the direction of this dissertation. He not only provided insightful feedback on my work but also offered suggestions for research methods and provided access to valuable resources. His guidance helped me develop my research skills and gain a deeper understanding of ways to bypass certain obstacles I encountered during the work process.

I would also like to show appreciation for the opportunity to develop this dissertation as part of the ”Lab4U&Spaces - Living Lab of Interactive Urban Space Solution” research grant project at Centro ALGORITMI, University of Minho. The resources, facilities, and support provided by the project were instrumental in successfully completing my research. I am thankful to the project team and everyone who contributed to it.

I also want to thank my friends, who have been an incredible source of support and encouragement throughout my academic journey. I am grateful for the countless hours spent discussing ideas, working together, and providing feedback. Their contributions have been invaluable and have significantly shaped the course of my academic journey. Through their presence, I have been able to develop skills and knowledge that have significantly shaped the course of my educational journey.

Finally, I want to express my deep appreciation to my family for being my constant support and inspiration throughout my educational career. Their love and support have given me the strength and determination to pursue my dreams and overcome the challenges that came with my academic journey. Without their encouragement and guidance, I would not be where I am today. Their unwavering belief in me has been a constant source of motivation, and I am forever grateful for their support.

DECLARAÇÃO DE INTEGRIDADE

Guimarães, de abril de 2023

(Henrique Miguel Cardoso Matos)

"You never fail until you stop trying."

Albert Einstein

Resumo

Nesta dissertação é explorado o tema da monitorização de múltiplos objetos no contexto de um ”smart campus”, com foco no contexto específico num campus universitário, sendo este o tema principal do projeto de investigação Lab4USpace. A monitorização de múltiplos objetos, especialmente de pessoas, é relevante para diversas aplicações, incluindo aplicações de vigilância, mobilidade e inteligência ambiental. No entanto, torna-se particularmente desafiante no contexto de espaços abertos, às quais exigem soluções com múltiplas câmaras com problemas inerentes, tais como a reidentificação.

O objetivo desta dissertação é desenvolver um *framework* capaz de fornecer informações sobre o percurso de várias pessoas ao longo do campus universitário usando um cenário com múltiplas câmaras. A solução visa não só a monitorização de uma pessoa num único cenário, mas também em todo o campus, coberto por diversas câmaras com ou sem sobreposição.

Esta dissertação discute os diversos desafios enfrentados durante o desenvolvimento deste projeto, incluindo preocupações com a privacidade e segurança dos utilizadores do campus. Com isso, optou-se por não enviar imagens para nenhuma aplicação, tratando apenas das informações estritamente retiradas da monitorização das pessoas. Um dos principais desafios foi desenvolver um framework que rastreie vários objetos num ambiente de um ”smart campus”, abordando desafios de espaços abertos e problemas de reidentificação. Além disso, devido aos recursos computacionais limitados, foi usado um computador de bordo para lidar com processamento de imagens e operações relacionadas às técnicas de visão computacional de maneira mais eficaz.

O framework proposto utiliza modelos de deteção de objetos e algoritmos de monitorização em tempo real que foram comparados neste contexto específico. Depois de pesquisar outras alternativas, a estrutura usa o modelo YOLOv7-tiny para deteção de objetos, BoT-Sort para a monitorização dos vários objetos e Deep Person Reid para a reidentificação. O programa foi desenvolvido em Python e juntamente a ele foi também criado um website para alterar as configurações do sistema de monitorização utilizando o framework Flask. Um message broker também foi utilizado para a comunicação entre os diversos componentes do sistema.

Os testes de validação demonstram a eficácia da framework proposta na monitorização das várias pessoas em todo o campus. O sistema proposto contribui significativamente para o desenvolvimento de soluções de múltiplas câmaras mais eficientes e eficazes para aplicações de ”smart campus”, com benefícios potenciais para a segurança, proteção e gestão do campus.

No geral, esta dissertação apresenta uma estrutura que rastreia de maneira eficaz várias pessoas num ambiente de ”smart campus”. A framework é uma contribuição importante para o desenvolvimento na área do ”smart campus” e tem potencial para desenvolvimento futuro e aplicações para além do campus universitário.

Palavras chave— Campus inteligente, Detecção de Objetos, Monitorização de Múltiplos Objetos, Reidentificação, Monitorização de Pessoas.

Abstract

This dissertation explores the topic of object multi-tracking in the context of a smart campus, focusing on the specific context of a university campus, being the main topic of the Lab4USpace research project. Multi-tracking of objects, especially people, is relevant for different applications, including surveillance, mobility, and ambient intelligence. However, it becomes particularly challenging in open spaces, which require multi-camera solutions with inherent issues like re-identification.

The objective of this dissertation is to develop a framework capable of providing information about the path of multiple people throughout the university campus using a multi-camera scenario. The solution aims not only to track a person in a single scenario but also over the entire campus, covered by various cameras with or without overlapping.

This dissertation discusses the challenges faced during the development of this project, including concerns about the privacy and security of campus users. As a result, the decision was made not to send images for any application, dealing only with the information strictly retrieved from the tracking. One main challenge was developing a framework that tracks multiple objects in a smart campus environment, addressing the challenges of open spaces and re-identification issues. Additionally, due to limited computational resources, an edge computer was used to handle image processing and computer vision-related operations more effectively.

The proposed framework uses different object detection models and real-time tracking algorithms that were compared in this specific context. After researching other alternatives, the framework uses the YOLOv7 tiny model for object detection, BoT-Sort for multiple object tracking, and Deep Person Reid for re-identification. The program was developed in Python and alongside it was also created a website to change the configurations of the tracking system using the Flask framework. A message broker was also used for communication between the various components of the system.

Validation tests demonstrate the effectiveness of the proposed framework in tracking multiple people across the campus. The proposed framework significantly contributes to developing efficient and effective multi-camera solutions for smart campus applications, with potential benefits for campus safety, security, and management.

Overall, this dissertation presents a framework that effectively tracks multiple people in a smart campus environment. The framework is an important contribution to the smart campus context and has the potential for future development and applications beyond the university campus.

Keywords— Smart Campus, Object Detection, Multiple Object Tracking, Re-Identification, People Tracking.

Contents

List of Figures

List of Tables

1 Introduction

The first chapter of the dissertation provides a broad introduction to the theme, the motivation behind the idea of this project, the objectives and goals to achieve with this specific task, the work plan, and the overall structure of the project, aiming to give a comprehensive understanding of the project focus and the various aspects that will be covered in subsequent chapters.

1.1 Context

As information and communications technology (ICT) becomes more powerful and computer processing power increases, new technologies are being developed at an unprecedented rate. One of the major areas of focus in this field is the capability of decision-making, which can potentially create smart systems capable of transforming different aspects of human life.

Smart cities have been trending due to the wide range of applications that can arise from their implementation, including traffic management, agriculture, parking, and waste management (Syed, Sierra-Sosa, Kumar, & Elmaghraby, 2021). All these applications are built around the paradigm of the Internet of Things (IoT), which involves the integration of a vast number of devices and sensors with low storage and processing capabilities, to improve the reliability, performance, and security of the smart city and its infrastructures (Arasteh et al., 2016).

A smart city is a city that utilises technology and data to become a fully efficient, monitored, and managed environment, capable of meeting the increasing needs of its citizens and addressing environmental goals and challenges (Fortes et al., 2019). However, implementing smart city applications and approaches requires extensive testing and study before they can be successfully deployed in the real world. Testing the changes in a smaller environment before releasing them to the public is often beneficial. Getting user feedback would also be a great way to investigate all the details. Although referring to the context of smart cities, which is more demanding given their potential size, the exact solutions can be equally beneficial in smaller spaces intended to be intelligent, as is the case in the smart campus. As a result, the concept of a smart campus resurfaced. It was in this logic of thought that the Lab4USpaces project emerged, which fits the work of this dissertation.

This dissertation focuses on tracking people through the university environment. This type of application is typically more focused on security and surveillance, but in this project, the goal is to contribute to developing smart space applications. Computer vision and artificial intelligence are essential to this project, as they enable the automatic extraction of useful information from a live camera, automatic detection, tracking, and recognition of objects of interest, and analysis of their activities (X. Wang, 2013).

However, specific challenges arise in implementing those smart space applications, such as the need for real-time processing and the accurate detection and tracking of individuals in crowded environments. This dissertation will explore these challenges and propose solutions to overcome them by developing an effective tracking system for a smart campus. This study aims to contribute to the development of smart spaces as a whole, paving the way for safer and more efficient urban environments.

1.2 Motivation

With the emergence of the latest pandemic virus, COVID-19, many restrictions had to be made to prevent the virus from spreading even more. One of them was the need to distance people from each other. With that in mind and with the help of the data generated from this program, the campus community could be informed about the risk they are undertaking. At the same time, security staff and managers could be alerted about potentially risky situations related to crowded spaces.

The original need led to new motivations, such as the desire to educate students about the extreme weather conditions, like UV radiation, they may encounter due to user carelessness.

With this kind of solution, a study can be made to understand the most common paths taken by the students, which can help improve the campus facilities. Additionally, the students could be informed about agglomerations or influx of people and prepare appropriate responses to avoid them.

As urban spaces become increasingly populated, it is essential to maintain high quality-of-life standards. Smart spaces are becoming increasingly important in this regard, and one of the primary goals of this study is to improve the quality of life on campus by optimising resources and guiding investments to enhance the overall campus experience.

1.3 Objectives

The main goal of this master's project is to develop a distributed tracking system that can track people throughout the university campus to create smart applications that inform students and managers of different types of adversities, addressing problems with management and mobility.

The specific objectives of this project are:

- To develop a distributed tracking system that can accurately detect and track individuals throughout the campus in real-time in low-performance devices.
- To understand the challenges and issues associated with introducing intelligence to a Smart Campus environment, particularly those related to user experience and privacy.
- To inform students of any agglomerations or dangers in the vicinity, providing them with timely information to take appropriate action.

To achieve these objectives, the following goals will be pursued:

- To identify appropriate models and algorithms for object detection, multiple object tracking, and re-identification to detect and track individuals throughout the campus in real-time accurately.
- To create an admin web page that can control the configurations of each tracking system, enabling the system to adapt to changing needs and conditions on the campus.

1.4 Methodology

During the development of this dissertation, the design science research (DSR) research method was used. It aims to generate knowledge of how things can and should be constructed or arranged to achieve a desired set of goals (Brocke, Hevner, & Maedche, 2020). Following the DSR paradigm, the research method can be organised in the most optimal way to find the best solution.

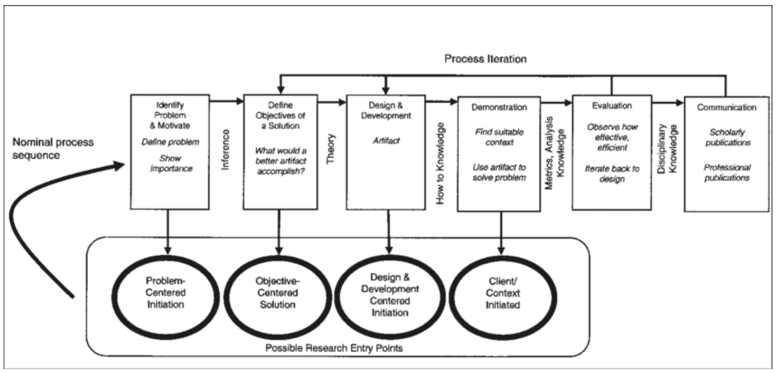

Process Iteration
Nominal process sequence
Identify Problem & Motivate
Define problem
Show importance
Inference
Define Objectives of a Solution
What would a better artifact accomplish?
Theory
Design & Development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artifact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary Knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development Centered Initiation
Client/ Context Initiated
Possible Research Entry Points

Figure 1: Design Science Research Methodology

The figure 1 shows the generic model the DSR method provides. It includes the following activities:

- *Problem identification and motivation*

  This activity defines a specific research problem and justifies the solution value. The purpose of justifying the value of a solution is twofold. It serves to motivate researchers to find solutions, as well as help the public evaluate the researcher's understanding of the problem.

- *Define the objectives for a solution*

  The solution objectives can come from defining the problem and knowing what is possible and what is viable. They can be quantitative, such as how much better the desired solution is than the current solution, or qualitative, such as describing how new models support a solution to an unsolved problem.

- *Design and development*

  From the previously defined goals, this activity involves determining the desired functionality and architecture of the system and then actually creating it.

- *Demonstration*

  This activity demonstrates the use of the system developed to solve one or more instances of the problem. This may include its use in experiments, simulations, case studies, evidence, or other appropriate activities.

- *Evaluation*

  Evaluation measures how well the developed system supports a solution to the problem by comparing it to actual observed results. At the end of this activity, can decide whether to return to step 3 to improve the system's effectiveness or to continue and leave further improvements for future projects.

- *Communication*

  The problem and all aspects of the designed system are communicated to the relevant stakeholders. Appropriate forms of communication are used depending on the research objectives and audience.

1.5 Dissertation Structure

The dissertation is organised into six chapters. Chapter 1 overviews the research context, motivation, and objectives. It also includes a discussion of the methodology used, the technologies employed, and the structure of the dissertation. Chapter 2, the State of the Art, focuses on the Internet of Things, smart spaces, computer vision, and related works. It provides a comprehensive review of the existing literature in these areas, including a discussion of the most recent developments and current challenges. Chapter 3, System Architecture Solutions, presents an analysis of the tracking system algorithms, a description of the edge configuration server, and a discussion of the proposed solution. Chapter 4 describes the design and implementation of the tracking system and the Edge Configuration Server, including a discussion of the software used. Chapter 5 presents the results of the program developed to evaluate the proposed solution, including a description of the setup, the tracking system, the edge configuration server, and the networking. Finally, Chapter 6 offers conclusions and recommendations for future research, including a discussion of potential avenues for future work based on the results of the case studies. Throughout the dissertation, figures, and tables supplement the text and provide visual representations of the data.

2 State of the art

This section will present an overview of the current state of the art of the Internet of Things, Smart Spaces, Computer Vision, and related projects. The focus is on exploring key concepts and reviewing related works by other researchers in the scientific community. This literature review aims to provide a comprehensive understanding of the field's current state by analysing the approaches and decisions made in similar projects. The research was conducted using various online resources such as *Google Scholar*, *IEEE Xplore*, *Papers With Code*, *ArXiv*, *Springer*, and *Google*.

2.1 Internet of Things

Since its initial introduction by Kevin Ashton in 1999 (Kramp, Kranenburg, & Lange, 2013), the Internet of Things (IoT) concept has gained popularity in both the majority of new applications being developed and in the daily lives of many people. IoT refers to using physical items with sensors, software, and other technologies to connect to and share data with other systems and devices through the Internet (*What is the internet of things (IOT)?*, n.d.).

Building an IoT system requires the development of three key components, as stated by (H. Wu, Han, Wang, & Sun, 2020):

- Sensing, identification and authentication technologies
- Transmission and communication technologies
- Data computing and processing technologies

A wide range of applications re-emerged with the development of the fundamental IoT components, as shown in figure 2 by (Challa et al., 2017).

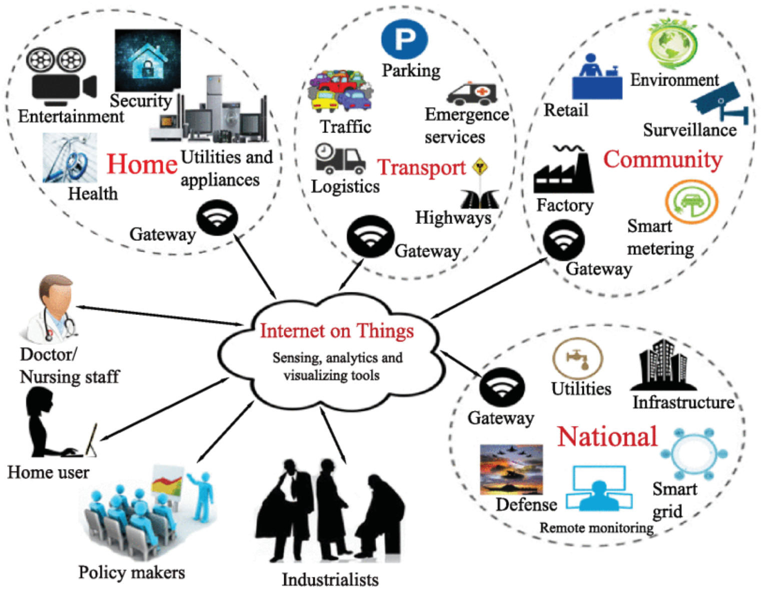


Figure 2: Internet of Things diagram showing different types of applications (Challa et al., 2017).

Despite many applications, several security concerns must be considered when building an IoT system, including challenges with reliability, access control, privacy, trust, safety, identification, and authentication (Challa et al., 2017). Different standards and protocols have been developed to ensure effective communication, consistent operation, and secure data transmission between devices and systems, as discussed in the following paragraphs (Gil, Ferrández, Mora, & Peral, 2016; K. Vaigandla, Radha, & Allanki, 2021; K. K. Vaigandla, Karne, & Rao, 2021).

Communication protocols used in IoT include Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), and Extensible Messaging and Presence Protocol (XMPP). MQTT is a lightweight, straightforward protocol suitable for devices with limited computing resources. CoAP is designed for low-power devices that need to communicate with the internet, while XMPP is a communication protocol used in instant messaging systems.

Wireless standards have also been developed to facilitate wireless communication between IoT devices, including Wi-Fi, Bluetooth, and Zigbee. Wi-Fi is a high-bandwidth standard suitable for high-speed data transfer devices. Bluetooth is a low-power wireless standard ideal for devices that communicate over short distances. At the same time, Zigbee is a low-power wireless standard designed for devices that need to communicate with other devices in a mesh network.

Security standards are critical in IoT to ensure secure data transmitted between devices. Some of the security standards used in IoT include Secure Sockets Layer/Transport Layer Security (SSL/TLS), Open Authorisation (OAuth), and Public Key Infrastructure (PKI). SSL/TLS provides secure communication between devices by encrypting data transmitted over the network. OAuth is used to authenticate and authorise access to web resources, while PKI offers secure transmission by using digital certificates to verify the identity of devices.

This dissertation is connected to the Internet of Things paradigm so that all data about the position of people on the university campus can be used to create applications that users can use for mobility, management, and comfort.

2.2 Smart Spaces

Smart spaces are physical environments that leverage advanced technologies like the Internet of Things (IoT) and Artificial Intelligence (AI) to create interactive and adaptive spaces. By integrating these technologies, smart spaces offer several benefits to individuals and organisations, including increased efficiency, improved safety, and enhanced user experience.

The rise of smart spaces results from the challenges cities face today. Economic turmoil, rapid urbanisation, climate change, and population growth have resulted in chaos and disorganisation. These problems, including health, transportation, pollution, waste management, and infrastructure issues, have hindered the development of cities (Joshi, Saxena, Godbole, & Shreya, 2016a). Smart cities aim to address these problems by harnessing the power of ubiquitous communication networks, highly distributed wireless sensor technology, and intelligent management systems (Clarke, 2013). With ICT and IoT technologies playing a significant role in the development of smart cities, smart spaces have become a crucial component.

With creating smart cities, (Joshi, Saxena, Godbole, & Shreya, 2016b) identified six crucial pillars: Social, Management, Economy, Legal, Technology, and Sustainability. Smart spaces are an integral part of the technology pillar and offer a practical solution for realising the vision of smart cities. For example, intelligent traffic management systems can optimise traffic lights, monitor traffic flows, and reduce congestion. Autonomous driving, enabled by smart spaces, can further enhance traffic management by allowing vehicles to communicate with each other and synchronise their paths to avoid collisions.

Overall, smart spaces are an essential element in the development of smart cities. Their ability to provide interactive and adaptive environments, coupled with the power of advanced technologies, offers immense potential for improving the quality of life in cities.

2.2.1 Smart Campus

A smart campus can be considered a small-scale city that utilises advanced intelligence technologies to enhance the overall experience for students, faculty, staff, and visitors. The main goal of a smart campus is to integrate various systems and technologies throughout the university to improve student performance, graduate quality, ease of life, and management. This includes the use of information technology services that are valuable, dynamic, and user-centred, enabling automation and real-time reporting support for a wide range of activities and systems, including (Muhamad, Kurniawan, Suhardi, & Yazid, 2017):

- Learning and education, such as e-learning, virtual and augmented reality, and personalised learning experiences.
- Social interaction, such as campus-wide communication systems and mobile apps for events and activities.
- Environments include building management systems, smart lighting and HVAC, and sustainable energy management systems.
- Office management, such as real-time space management, booking and scheduling, and facilities management.
- Energy saving includes energy-efficient building design, monitoring and control systems, and integration of renewable energy.
- Safety and security include automated security and surveillance systems, access control and monitoring systems, and emergency response systems.
- Transportation includes real-time parking and transportation management systems and smart traffic control systems.
- Health and well-being include fitness tracking, healthy eating options, and mental health support.

To support all these services, the smart campus uses a variety of network technologies, including:

- Ethernet: This widely used network technology supports wired device connectivity.
- Wi-Fi: This wireless networking technology allows devices to connect to the internet and each other without needing physical cables.
- Cellular: This technology allows devices to connect to the internet and each other using cellular networks, such as 4G and 5G.
- Zigbee: This wireless networking technology is designed for low-power, low-data-rate applications like sensors and other IoT devices.
- Bluetooth: This wireless networking technology is commonly used for short-range communications between devices, such as smartphones and wireless headphones.
- LoRaWAN: This low-power wide-area network (LPWAN) technology is designed for IoT devices that need to connect to the internet from long distances.
- RFID: Radio-frequency identification (RFID) is a wireless technology that uses radio waves to transfer data between devices and is widely used to track and identify assets and people.

- MQTT: MQTT (Message Queuing Telemetry Transport) is a publish-subscribe messaging protocol designed for IoT devices with limited resources. It is lightweight and efficient, making it well-suited for smart campus applications.

By using these network technologies, a smart campus can create a robust and flexible network infrastructure that allows for seamless communication and data transfer between all systems and devices. This enables real-time monitoring and control, as well as the ability to analyse data and make decisions to optimise the performance of the campus (Abuarqoub et al., 2017; Alghamdi & Shetty, 2016).

2.3 Computer Vision

Computer vision is a sub-field of artificial intelligence (AI) that enables machines to extract meaningful information from visual sources such as digital photos and videos and recommend or perform actions in response to that information. AI gives computers the ability to reason, whereas computer vision gives them the ability to see, observe and understand.

While human vision and computer vision share some similarities, humans have the advantage of years of training and experience that allows us to discern between objects, judge distances, and perceive motion, among other things. On the other hand, computer vision relies on cameras, data, and algorithms to perform similar functions (*What is Computer Vision?*, n.d.).

Computer vision contains different subdomains such as scene reconstruction, object detection, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, visual servoing, 3D scene modelling, and image restoration (*Computer vision*, 2023).

In this dissertation project, specific sub-areas of computer vision were used. The subsequent subsections provide a more detailed overview of these areas.

2.3.1 Object Detection

Object detection is a fundamental task in computer vision, which involves locating and classifying objects in images or videos. This task can be further divided into sub-tasks such as pedestrian detection, vehicle detection, and skeleton detection, among others. A popular dataset used for training and evaluating object detection models is COCO (Common Objects in Context), which contains 91 object types and 2.5 million labelled instances in 328k images (Lin et al., 2014).

Methods used for object detection can be broadly categorised into two approaches: neural network-based and non-neural. While non-neural techniques have been used in the past, recent advances in processing power and deep learning have made neural network-based systems the dominant method for object detection. Some of the most commonly used deep neural network architectures for object detection include Faster R-CNN, YOLO (You Only Look Once), and SSD (Single Shot MultiBox Detector) (Zou, Shi, Guo, & Ye, 2019).

With the advances in detection technologies, two object detectors have resurfaced: single-stage detectors and two-stage detectors. Single-stage detectors, such as YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector), approach object detection as a straightforward regression problem. These models take an input image and directly output the class probabilities and bounding box coordinates for each object in the picture. The advantage of single-stage detectors is their speed, as they only require one pass through the neural network for inference. However, the downside is that they may sacrifice some accuracy for speed, as they have fewer stages to refine their predictions. Two-stage detectors like Faster R-CNN (Region-based Convolutional Neural Networks) or Mask R-CNN take a different approach. First, they generate a set of object proposals using a Region Proposal Network (RPN), which identifies regions in the image likely to contain objects. These regions are then fed into a second stage that performs classification and bounding-box regression to refine the object proposals. Two-stage detectors typically have higher accuracy than single-stage detectors, with more steps to refine their predictions. However, they are slower than single-stage detectors, as they require multiple passes through the neural network for inference (Soviany & Ionescu, 2018).

Although object detection can produce some fascinating results, there are several challenges involved (Zou et al., 2019):

- *Small pedestrian*

  This happens when pedestrians are caught away from the camera and have a low number of pixels to detect correctly.

- *Hard negatives*

  Some objects in the environment can be very similar to pedestrians.

- *Dense and occluded pedestrians*

  Pedestrians can go unnoticed by being obscured by other pedestrians or objects. Sometimes they may only be partially obscured, making discovery unlikely.

- *Real-time detection*

  Real-time detection is essential for some applications, such as autonomous driving and video surveillance. Hence the detection time must be very short.

Later in this dissertation, some object detection algorithms are compared to find the most appropriate model for this project.

2.3.2 Multiple Object Tracking

Multiple object tracking (MOT) is a requirement in computer vision, especially when looking at large spaces. It is also known as multiple target tracking (MTT), which aims to track objects in a video sequence while maintaining their identity. For example, objects can vary from pedestrians or vehicles on the street, athletes on the field, or groups of animals. Due to the variety of objects of interest, many applications have emerged, such as video surveillance, autonomous driving, mobile robotics and virtual reality (Luo et al., 2021).

There are two different kinds of trackers: online tracking, also known as sequential tracking, which handles the frames in a step-by-step manner and can be used for real-time tracking, and offline tracking, which processes a batch of frames from the video to produce a more accurate prediction but cannot be used for real-time tracking.

During the video tracking, multiple errors can occur (Luo et al., 2021):

- Frequent occlusions
- Initialisation and termination of tracks
- Similar appearance
- Interactions among multiple objects

To compare the models in multiple object tracking methods, evaluation is necessary. Standard metrics include concepts like False negatives (FN), False positives (FP), Fragmentation, Mergers (ID Switch), and Deviation. Using these general concepts creates metrics, such as Multiple Object Tracking Accuracy (MOTA), which considers three sources of errors with a single performance measure, false negatives, false positives and identity switches. Figure 3, which was taken from (Dendorfer et al., 2020), illustrates how some frequent mistakes might occur.

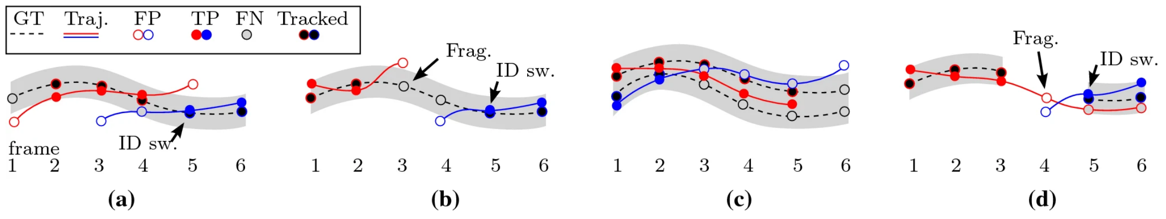


Figure 3: Common errors in MOT (Dendorfer et al., 2020).

Building an MOT algorithm that can track objects precisely and accurately requires considering several computational strategies. However, before employing any method, we must first determine the primary goals: evaluating object similarity across frames or recovering the identity information using object similarity. The first relates to modelling appearance, motion, interaction, exclusion, and occlusion, and the second refers to the inference problem. An overview of all the MOT components is provided next.

- *Appearance*

  Most MOT algorithms do not acknowledge this as the core component, but it can be essential. It may be separated into two categories: statistical measurement, which computes the affinity between two observations based on the visual characteristics recovered, and visual representation, which extracts local and regional data such as optical flow, covariance matrix, colour features, and depth.

- *Motion*

  This model captures the dynamic behaviour of the objects, and it tries to estimate the position of the objects in future frames. Two types of models are used linear and non-linear motion.

- *Interaction*

  This captures how one object can affect other objects. An illustration of social force would be when a person must alter their speed, direction, and destination to prevent collisions. An example of crowd motion would be when individuals follow and guide one another in a crowded area.

- *Exclusion*

  Exclusion is a restriction used to prevent physical collisions; it results from the fact that two different objects cannot occupy the same space in the actual world.

- *Occlusion*

  Occlusion is one of the major obstacles to object tracking and can lead to identity switches and trajectory fragmentation. There are several methods to solve this kind of issue, including Part-to-whole techniques, which operate under the premise that since some portion of an item is still visible, the condition of the object as a whole may be deduced;

The technique, known as Hypothesise-and-test, addresses occlusion by generating hypotheses and evaluating them according to observations at hand; Buffer-and-recover is a technique that stores information about the state of the objects before occlusion occurs. When occlusion ends, the states are retrieved using the buffered observations and the previously saved forms.

- *Inference*

  Two approaches can be considered when discussing inference methodology: probabilistic inference, where the goal is to estimate the target state probabilistic distribution using various probability reasoning techniques based on past and present observations, making it suitable for online tracking. Several probabilistic inference models, including the Kalman filter, Extended Kalman filter, and Particle filter, are available in MOT. Deterministic optimisation is another method better suited for offline tracking since it requires observations from every frame or time window to achieve its primary objective of finding the optimal association. Numerous strategies exist, including maximum-weight independent set (MWIS), conditional random field, bipartite graph matching, dynamic programming, and min-cost max-flow network flow.

All the material discussed in this part was taken from books (Dendorfer et al., 2020), (Luo et al., 2021), and (Zheng et al., 2021), further clarification and information can be found in these articles.

2.3.3 Re-Identification

Re-identification is a big issue in computer vision research that has gained more attention recently because they play a key role in surveillance systems and public security applications, being able to prevent crimes and terrorist activities and can help in forensic investigations (Almasawa, Elrefaei, & Moria, 2019). This is defined as the process of seeking the occurrences of a query person from a set of gallery candidates, where the images are captured from different non-overlapping camera views.

Re-ID is a highly challenging task because of the apparent discrepancies that might occur, such as the viewpoint, illumination, resolution, occlusion, colour, background, etc. (Leng et al., 2020).

The two primary trends in re-identification techniques are closed-world and open-world. In the following paragraphs, both approaches are compared (M. Ye et al., 2020):

- Single-modality vs Heterogeneous Data

  For raw data collection, all individuals are represented by images/videos captured by single-mode view cameras in the closed world. However, in practical open-world applications, we may also need to process heterogeneous data, such as infrared images, sketches, depth images, or textual descriptions.

- Bounding Box Generation vs Raw Images/Videos

  For the bounding box generation, the closed-world person Re-ID typically conducts training and testing using produced bounding boxes, which are mostly made up of information about a person's look. On the other hand, specific real-world open-world applications demand a complete person search from unprocessed photos or videos.

- Sufficient Annotated Data vs Unavailable/Limited Labels

  For the training data annotation, the closed-world person Re-ID usually assumes that we have enough annotated training data to train the supervised Re-ID model. However, annotating the tag for each pair of cameras in each new environment is time-consuming, labour-intensive, and very expensive. We may not have enough annotated data or even labelled information in open-world scenarios.

- Correct Annotation vs Noisy Annotation

  For model training, existing closed-world person Re-ID systems usually assume that all the annotations are correct, with clean labels. However, annotation noise is usually unavoidable due to errors or poor detection/tracking results.

- Query Exists in Gallery vs Open-set

  In the pedestrian retrieval stage, most existing closed-world person Re-ID works assume that the query must occur in the gallery. However, in many scenarios, the query person may not appear in the gallery set, meaning verification must be performed rather than retrieved.

It is clear after analysing that the open-world scenario is more appropriate for this dissertation work. Figure 4, retrieved from (Leng et al., 2020), illustrates how the re-identification works.

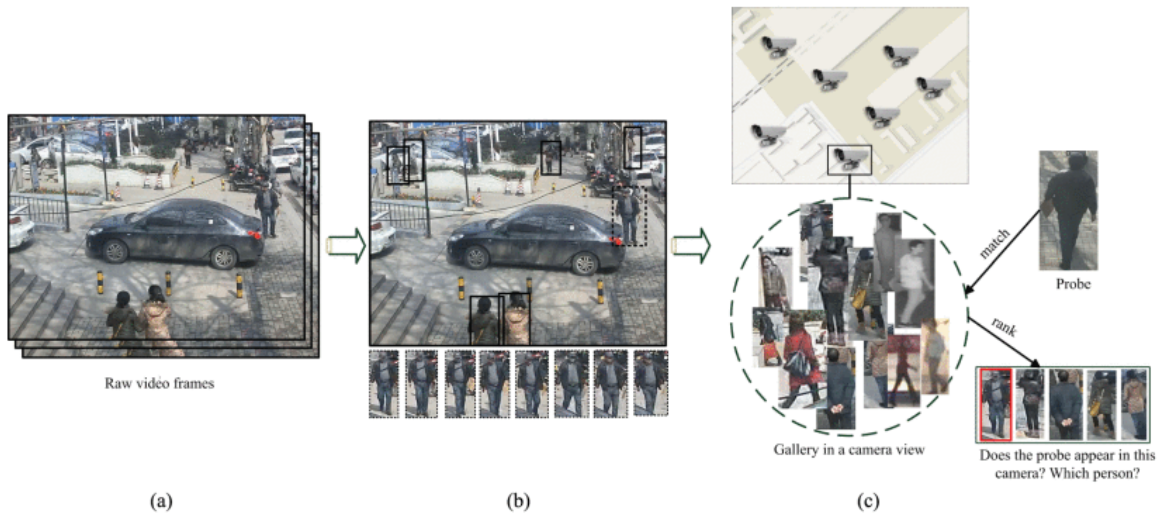


Figure 4: Example of how re-identification operates. (Leng et al., 2020)

2.4 Related projects

This section discusses several related projects and how they complement or compare to the focus of this dissertation.

The first project discussed is related to computer vision techniques in a smart campus environment using an automated parking system described in the paper by Banerjee, Ashwin, and Guddeti (2019). The system uses cameras and image processing algorithms to detect available parking spaces and guide vehicles to the nearest empty spot. It also keeps track of occupied areas, improving parking management efficiency and reducing the time and effort required to find a parking spot.

This paper details the challenges faced during the development of the system, including image processing techniques and system architecture. The results demonstrate the effectiveness of the proposed method in improving parking management in a smart campus environment. The contribution of this paper lies in the practical application of computer vision techniques in a real-world scenario and its potential in other parking systems in smart cities.

Although the automated parking system project focus differs from the object multi-tracking framework proposed in this dissertation, both projects highlight the potential of computer vision techniques in smart campus environments. Furthermore, both projects demonstrate the importance of addressing challenges related to image processing techniques, system architecture, and privacy and security concerns in developing smart campus solutions.

The paper "A Low-Cost IoT Cyber-Physical System for Vehicle and Pedestrian Tracking in a Smart Campus" (Toutouh & Alba, 2022) presents a low-cost IoT system that tracks vehicles and pedestrians in a smart campus environment. The paper addresses the importance of improving safety and security in smart campuses, as these environments are often open and vulnerable to security threats.

The authors proposed a system composed of IoT devices, such as sensors and cameras, that collect data on the location and movement of vehicles and pedestrians. This data is then processed by a cloud-based server and displayed in real-time through a user interface. The system has shown effectiveness in providing real-time information on the location and movement of vehicles and pedestrians, reducing the risk of security incidents.

While the proposed system in the paper has a different focus than this dissertation, both projects share a similar goal of improving safety and security in smart campus environments. In contrast to the paper, this dissertation explicitly addresses the challenge of multi-people tracking using distributed sensor networks. However, both projects face similar challenges related to data privacy and security.

The paper proposes solutions such as data encryption and secure communication protocols to address these challenges, which can also be relevant to this project. The paper also demonstrates the potential for low-cost IoT solutions to improve safety and security in campus environments. This insight can be suitable for developing this project, as using an edge computer is an attempt to address limited computational resources, which is also a concern for low-cost IoT solutions.

The paper "Toward a smart campus using IoT: Framework for safety and security system on a university campus" (Alghamdi, Thanoon, & Alsulami, 2019) presents a framework for a smart campus that aims to improve the safety and security of a university campus using IoT devices. The authors propose a multi-layer architecture that includes physical, communication, and application layers. The physical layer consists of various IoT devices, such as sensors and cameras, that collect data about the campus environment. The communication layer transfers the data to a central server for processing and analysis. In contrast, the application layer provides a user interface for administrators to monitor and respond to potential security incidents. This paper is highly relevant to this dissertation project, as it also aims to improve the safety and security of a university campus. However, while the paper focuses on integrating various IoT devices to collect data about the campus environment, this project focuses on tracking people using a multi-camera scenario.

The paper presents a comprehensive framework that addresses the challenges of data privacy and security in implementing a smart campus. The authors propose solutions such as data encryption and secure communication protocols to ensure the safety of the collected data. This is an important consideration for this project, as it also needs to address concerns about the privacy and security of campus users.

Overall, the paper provides valuable insights into implementing a smart campus framework. While this dissertation takes a different approach, the paper's emphasis on data privacy and security can inform the development of this project framework.

The paper written by M. Wu, Qian, Wang, and Yang (2021) presents a system for tracking vehicles between multiple cameras in a city to regulate traffic. The system uses a combination of vehicle re-identification (Re-ID), single-camera tracking (SCT), and multi-camera tracks matching (MCTM) to track vehicles accurately across a city-wide network of cameras.

To achieve this, the authors compare several object detection models and ultimately choose EfficientDet for their system. They also use DeepSort for object tracking and employ ResNet with IBN structure and non-local module for the re-identification model backbone, generalised-mean (Gem) for the aggregation, and bnneck for the head. Additionally, they developed a technique for spatial attention by eliminating the background of the bounding boxes. To train their models, the authors use the VeRi776 dataset. Their system achieves high levels of precision in vehicle tracking, which has important implications for traffic regulation and management in urban areas.

While this paper focuses on vehicle tracking, the techniques and methodologies presented can be applied to other areas, including tracking people in a smart campus environment. Using multi-camera tracking and object re-identification, similar frameworks can be developed to track individuals across a wide area, providing valuable data for campus management, security, and safety.

In the paper by Hu, Hachiuma, Saito, Takatsume, and Kajita (2022), the authors present a novel system for multi-camera multi-person tracking and re-identification in an operating room. Their approach aims to analyse the surgical progress by tracking the movements of multiple individuals inside the room using overlapping cameras. This requires inter-camera re-identification, which combines pedestrian detection with pose estimation and monocular tracking.

The authors used the AlphaPose and YOLOv3 models to recognise pedestrians with pose estimation. They also developed a method inspired by Trackpy for monocular tracking. However, the most innovative aspect of their work is the clustering algorithm they designed for inter-camera re-identification. They used the density-based spatial clustering of applications with noise (DBSCAN) algorithm based on grouping nearby data points with similar features.

While their work focuses on a specific application domain, i.e., an operating room, it demonstrates the potential of multi-camera tracking systems that rely on inter-camera re-identification. The author's use of overlapping cameras and their clustering algorithm makes it possible to track the movements of multiple individuals, even in complex environments with occlusions and other challenges. Compared to this dissertation, the paper has a more specific focus and application domain. However, it provides valuable insights and technical solutions that can be adapted to other settings and use cases. For example, their use of multiple cameras and inter-camera re-identification could be helpful in scenarios where a more expansive area needs to be monitored, such as public spaces or transportation hubs.

The paper by J. Ye et al. (2021) presents a robust multi-target multi-camera (MTMC) tracking system for traffic flow analysis and vehicle route planning. Compared to this dissertation project, their focus is more on vehicle tracking and traffic analysis than on security and surveillance of people. They employ various methods to achieve their objective, including vehicle detection, re-identification, single-camera tracking, and inter-camera association.

For vehicle detection, the authors use the Cascade R-CNN network with a ResNet-101 backbone equipped with the feature pyramid network (FPN) for feature extraction. Meanwhile, the vehicle re-identification model used is HRNet with Res2Net as the backbone. The appearance features extracted from these models are concatenated to create a more accurate representation.

To track multiple targets within a single camera view, they utilise the Tracklet-Plane Matching (TPM) method, known for its efficiency and high performance. Finally, they provide an overview of different ways to enhance inter-camera association, including identifying enter and exit areas of trajectories, distinguishing and filtering out candidate trajectories based on traffic rules, road structures, and travel time, as well as comparing trajectories between multiple cameras.

While their paper differs from this dissertation in terms of focus and objective, there are similarities in the methods used, such as deep learning-based models for object detection and re-identification. Their paper offers valuable insights into the field of multi-camera tracking and association for traffic analysis and route planning, and their methodology may provide helpful inspiration for future research in related fields.

In the context of people tracking, the paper ”People Tracking System Using DeepSORT” (Azhar, Zaman, Tahir, & Hashim, 2020) presents a real-time single-camera system designed for crowd surveillance. The paper utilises YOLOv3 for object detection and Deep SORT for object tracking to track people even in the event of occlusion. However, the models used in this paper are outdated compared to more recent ones.

While this paper addresses the topic of people tracking, it is limited to a single-camera system and focuses on tracking individuals in crowded environments. In contrast, the proposed framework in this dissertation focuses on multi-people tracking in a smart campus environment using distributed sensor networks. Moreover, the proposed framework utilises advanced object detection and tracking algorithms such as BoT-Sort and Deep Person Reid for re-identification, making it more effective in tracking multiple individuals across the campus.

The proposed framework in this dissertation also addresses the challenges of open spaces, re-identification, and the privacy and security of campus users, which are not discussed in the paper. Therefore, the proposed framework presents a more comprehensive solution for multi-people tracking in a smart campus environment.

In their paper, ”Features for Multi-Target Multi-Camera Tracking and Re-Identification,” (Ristani & Tomasi, 2018) present a system that aims to locate the position of every individual present in a real-time streaming video using multiple cameras. The project objectives are similar to this dissertation, as both aim to develop a tracking framework capable of tracking various objects, particularly people, in a complex, multi-camera environment.

To achieve their goal, the authors of the paper divided their tracking system into different components, including person detection, data association, appearance, multiple cameras, and learning to track. They used various algorithms and techniques to develop their tracking system, called DeepCC, for the Multi-Target Multi-Camera Tracker (MTMCT). DeepCC combines different algorithms for data augmentation, motion correlation, optimisation, and multi-level reasoning with OpenPose as the person detector and ResNet50 as the appearance feature extraction engine. In addition, they developed a re-identification model called Adaptive Weighted Triplet Loss (AWTL) to accurately identify individuals across multiple cameras.

While the purpose and approach of the related project are similar to this dissertation, the two have some differences. For instance, this dissertation focuses specifically on multi-object tracking in the context of a smart campus, whereas the related project addresses a broader range of applications. The corresponding project also uses an older re-identification model compared to the more recent Deep Person Reid model used in this dissertation. Nonetheless, the related project techniques and approaches can serve as a valuable reference for this dissertation and provide a basis for comparing and evaluating the proposed framework performance.

3 System Architecture Solution

After exploring the literature, it is evident that there are limitations on both the system and the component modules. The project goal is to install several cameras throughout the campus. Therefore one major concern is the security and privacy of campus users. It would be problematic if some people do not want their images to be used in applications that could result from this research. To address this issue, no images will be included in the final information for the applications. They will only be utilised to enable configuration and tracking in each system.

The tracking system performance is another major issue because it is stored in a container that is as compact as it can be and is located in a difficult-to-access location, like the top of a street light. Therefore, the tracking system computer must be as small as possible. Also, a GPU might not be the better option due to space and energy consumption constraints, although it can significantly boost the performance of each model. Thus, a low-power fanless single CPU system is desirable for the tracking system.

3.1 Tracking algorithms analysis

To determine the best technologies, the fundamental models for each modular system's functioning will be compared in this subsection. Several tests detailed in the upcoming chapters were carried out using an Intel Core i7-8550U CPU running at 1.80GHz and 8GB of RAM machine (at the prototype level). Additionally, a GPU is used in some cases to extract statistics, indicating the use of an NVIDIA Tesla V100.

3.1.1 Identification of Objects

This section will focus on comparing and selecting the most suitable object detection model for this project. The evaluation will be conducted on a range of pre-trained models using the COCO (Common Objects in Context) dataset.

To identify the optimal model, a trade-off analysis will be conducted based on two primary metrics: COCO mean average precision (mAP), which measures the model effectiveness in detecting objects, and the average processing time required for each detection. The comparison results for each model can be found in table 1. The ultimate objective is to identify a model that offers the best balance between accuracy and speed, ensuring an efficient and reliable object detection performance for the project.

The initial set of models used in this project was sourced from the TensorFlow 2 collection, which features popular models such as CenterNet, SSD MobileNet, EfficientDet, and Faster R-CNN (Akkas, Maini, & Qiu, 2019) [1]. The YOLOv3 model, an upgraded version of YOLOv2 originally developed using the darknet, was also tested. It is available as an open-source neural network framework written in C and CUDA (Redmon & Farhadi, 2018) and was used in PyTorch format in this project [2]. YOLOv5, which operates using the PyTorch framework (Gai, He, & Zhou, 2021), was released in 2020 and ranges from nano to xlarge models with varying accuracy and processing times [3]. Another option tested was YOLOR (You Only Learn One Representation), an improved version of YOLOv4 (Bochkovskiy, Wang, & Liao, 2020), and Scaled-YOLOv4 (C.-Y. Wang, Bochkovskiy, & Liao, 2021). It features several models, and the tested models used the Cross-Stage-Partial (CSP) Network, allowing the network to

[1] `https://tfhub.dev/tensorflow/collections/object_detection/1`

[2] `https://github.com/ultralytics/yolov3`

[3] `https://github.com/ultralytics/yolov5`

Table 1 Models Comparison

Model Name	**COCO mAP (%)**	**GPU average process time (ms)**	**CPU average process time (ms)**
SSD MobileNet V2 FPNLite 320×320	22.2	22	227
Mobile Net V2 FPNLite 640×640	28.2	39	431
Faster R-CNN ResNet50 V1 640×640	29.3	53	2145 (Had memory problems)
Faster R-CNN ResNet101 V1 640×640	31.8	55	2490 (Had memory problems)
SSD ResNet50 V1 FPN 640×640 (RetinaNet50)	34.3	46	1396
YOLOv3 320×320	28.2	22	562
YOLOv3 608×608	33	50	890
YOLOv5n 640×640	28.0	6.3	**91**
YOLOv5s 640×640	37.4	6.4	176
YOLOv5m 640×640	45.4	8.2	424
YOLOR-CSP 640×640	52.8	9.4	577
YOLOR-CSP-X 640×640	**54.8**	11.5	967
YOLOv7-tiny 640×640	38.7	**3.5**	97.1
YOLOv7 640×640	51.4	6.2	463
YOLOv7X 640×640	53.1	8.7	1227

adjust its depth, width, resolution, and structure while maintaining speed and accuracy [4]. Finally, YOLOv7, the most recent of the models tested, is said to surpass YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, and other object detectors in terms of speed and accuracy (C.-Y. Wang, Bochkovskiy, & Liao, 2022) [5].

Looking into the results obtained shown in Table 1, it is clear that models with higher mAP tend to have longer processing times. However, the ideal model for this project should have high accuracy and a short processing time. For example, while YOLOv5n has a low mAP, it has a short processing time, and YOLOv5s has significantly higher accuracy than YOLOv5n despite a slightly longer processing time. YOLOv5m, YOLOR-CSP, and YOLOv7 all offer excellent accuracy but have long processing times. The best option for this project, with an average precision of 38.7% and a short enough average processing time of 97.1ms when using a CPU, is YOLOv7-tiny. If a GPU is available, models with higher mAP can be considered since processing time is always smaller. In that case, two models can be used: YOLOR or YOLOv7.

3.1.2 Tracking Multiple Objects

Once the best model for object detection is determined, the next step is to identify an appropriate multiple object tracking algorithm. In this study, four models are presented and compared. This work contrasts DeepSORT (Wojke, Bewley, & Paulus, 2017; Wojke & Bewley, 2018; Azhar et al., 2020), an update from the original SORT (Simple Online and Real-time Tracking) algorithm (Bewley, Ge, Ott, Ramos, & Upcroft, 2016) with an incorporated deep appearance descriptor. DeepSORT was introduced in 2017 and upgraded in 2018, but other algorithms have surpassed its accuracy, such as StrongSORT (Du, Song, Yang, & Zhao, 2022), an improvement of DeepSORT. StrongSORT was introduced in 2022 and ranked among the top three most accurate algorithms using the MOT17 dataset.

The final algorithm, BOT-SORT (Aharon et al., 2022), was developed in 2022 on top of a baseline model, ByteTrack (Zhang et al., 2021). BOT-SORT employs a series of algorithms, including an updated Kalman filter (KF), camera motion compensation (CMC), output tracks prediction (Pred), and with or without the ReID module. Among the most accurate models, BOT-SORT ranks second in the MOT17 and MOT20 datasets.

Tables 2 and 3 compare various MOT algorithms using as metrics HOTA (Higher Order Tracking Accuracy), IDF1 (ratio of correctly identified detections over the average number of ground-truth and computed detections) (Ristani, Solera, Zou, Cucchiara, & Tomasi, 2016), MOTA (Multiple Object Tracking Accuracy), and processing time per frame.

Although SORT has the fastest processing time per frame, its accuracy is unsatisfactory. While DeepSORT and StrongSORT perform well in actual tracking, they show a long processing time, particularly with the MOT20 benchmark. Therefore, BOT-SORT is the best option due to its quick processing time and high accuracy, particularly with MOT20 benchmark.

[4] `https://github.com/WongKinYiu/yolor`

[5] `https://github.com/WongKinYiu/yolov7`

Table 2 MOT17 Benchmark Comparison Aharon et al. (2022)

Algorithm	**HOTA**	**IDF1**	**MOTA**	**Frame Process Time (ms)**
SORT	34	39.8	43.1	**7**
DeepSORT	61.2	74.5	78	72.5
StrongSORT++	64.4	79.5	79.6	140.8
BoT-SORT	64.6	79.5	**80.6**	151.5
BoT-SORT-ReID	**65.0**	**80.2**	80.5	222.2

Table 3 MOT20 Benchmark Comparison Aharon et al. (2022)

Algorithm	**HOTA**	**IDF1**	**MOTA**	**Frame Process Time (ms)**
SORT	36.1	45.1	42.7	**17.5**
DeepSORT	57.1	69.6	71.8	312.5
StrongSORT++	62.6	77	73.8	714.3
BoT-SORT	62.6	76.3	77.7	151.5
BoT-SORT-ReID	**63.3**	**77.5**	**77.8**	416.7

3.1.3 People Re-Identification

The re-identification module is the final component of the tracking system, responsible for identifying individuals across different cameras. Several re-identification algorithms, such as centroids-reid (Wieczorek, Rychalska, & Dabrowski, 2021) and LUPerson (Fu et al., 2020), have been developed with high accuracy on specific datasets that have very predictive flows and shapes. However, these algorithms are not suitable for this project since the images used are more dynamic, and they all have similar requirements for time performance and computational resources.

This dissertation uses the Deep Person Reid method (Zhou & Xiang, 2019; Zhou, Yang, Cavallaro, & Xiang, 2019, 2021), published in 2019 and uses the torchreid library. The library provides various models to choose from, including shufflenet, mobilenetv2, and osnet. Deep Person Reid ranks 8 and 48 in the world when utilising the CUHK03 and Market-1501 datasets, respectively, although both datasets were used for training and inference.

The gallery and query dataset will not be stable in the real-world application of this project since there will be variations in lighting, posture, colours, and other properties. Therefore, a robust model is required that can handle different image variations. Deep Person Reid is an excellent choice since it is the top-performing method when trained and used in cross-domain with MSMT17 and Market-1501 datasets, which have similar characteristics to this project needs. Additionally, Deep Person Reid comes with extensive documentation [6] [7].

[6]`https://kaiyangzhou.github.io/deep-person-reid/user_guide`

[7]`https://github.com/KaiyangZhou/deep-person-reid`

3.2 Edge Configuration Server

Some configurations must be made for the tracking system to function as intended. These settings may provide new practical functions while ensuring the system correct operation. This server was created to simplify the process of changing the system configurations. Instead of physically accessing the machine or using a remote connection, like SSH, to make changes, the configurations can be easily accessed and modified through the website. Additionally, there is no need to manually restart the program for the changes to take effect. Several useful features are as follows:

- Check if the tracking system is currently running, particularly when multiple devices are in use, so it is easy to determine which ones are active.
- Viewing settings, such as the model and source being used on each device, ensure they are all configured correctly.
- Viewing and adjusting detection, tracking, and re-identification settings in real-time without restarting the program.
- Changing global coordinates to create a unified coordinate system that considers the scale, angle, and offset of each device and can be used to obtain final tracking positions throughout the campus.
- Creating zones of interest in the target scenario, such as specific areas where people are expected to congregate, to provide extra information about traffic patterns.
- Removing areas from the frame that are irrelevant to the tracking system, such as areas where no people are expected to be present and in the case of two cameras overlapped in one of them, remove the overlapped area.
- Creating lines in the frame, which can be used to define specific names for each crossed direction and obtain information about how many people travel in a particular direction. Not only that, but this configuration is also necessary to create an association between tracking systems.
- Creating re-identification associations between cameras enables the system to track people as they move between devices and identify them as the same individual.

3.3 Proposed Solution

This dissertation project is based on a four-layer architecture of the Internet of Things (IoT) as shown in Figure 5, which addresses the challenges of scalability, security, and interoperability commonly faced in traditional IoT architectures. The four layers are the Physical layer, the Network layer, the Integration layer, and the Application layer. Each layer serves a specific purpose and plays a unique role in the overall functioning of the architecture. The Physical layer is responsible for the physical devices that make up the IoT system, such as sensors and actuators. These devices are connected to the Network layer, responsible for the communication and data transfer between the devices. The Integration layer provides the necessary services for device management, data storage, and analytics. Finally, the Application layer allows for the creation of user-interfacing applications that use the data collected by the devices.

In this IoT architecture, Service-Oriented Architecture (SOA) is a fundamental design principle, although not explicitly shown in Figure 5. SOA supports the architecture's goals, enhancing scalability, security, and interoperability. It fosters modular, independent services within the Integration and Physical layer, enabling efficient device, data, and analytics management. This approach emphasises loose coupling and interoperability among services, aligning with broader objectives and addressing traditional IoT architecture challenges.

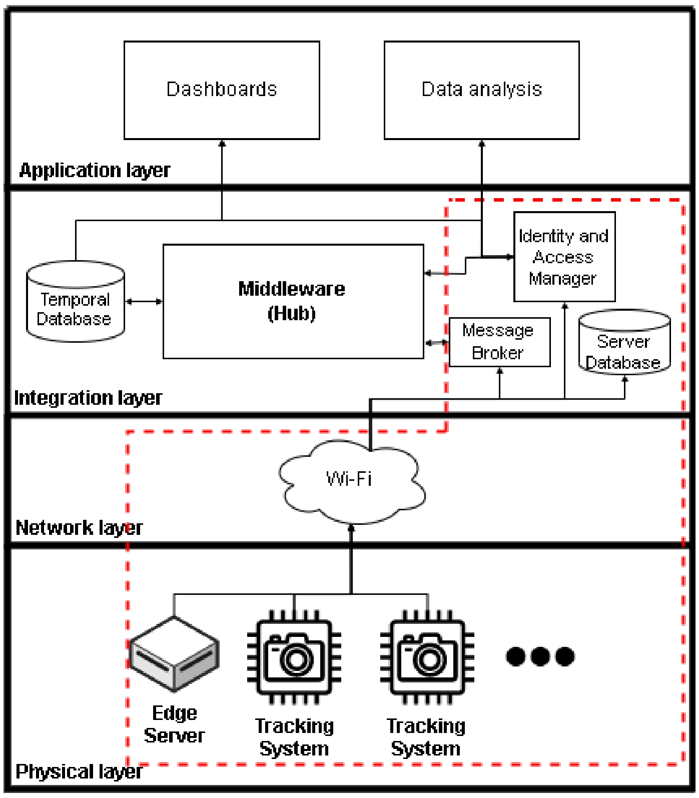


Figure 5: Global system architecture.

The details of Figure 5 show all the modules in each layer for this MOT solution to work as intended. Firstly, the physical layer contains the tracking systems (IP intelligent cameras) and an edge server that manages their configuration and data normalisation concerning the object's geolocation. The network layer enables communication between the tracking systems and the middleware using Wi-Fi or similar network technology. The integration layer includes the middleware, which works as a hub, an identity and access manager for device authentication and general access control functions, a message broker for communication organisation, a temporal database for data storage, and another database to store the server configurations during run time. The collected data is used in the application layer for analysis, visualisation, and other decision-support applications. In the image, a stripped red line can be seen, evidencing this dissertation project's main targets.

In figure 6, is shown an alternative depiction of the architectural framework, focusing exclusively on the intricacies of this dissertation's architecture. This representation shows the network of connections that interlink all the essential components within the system.

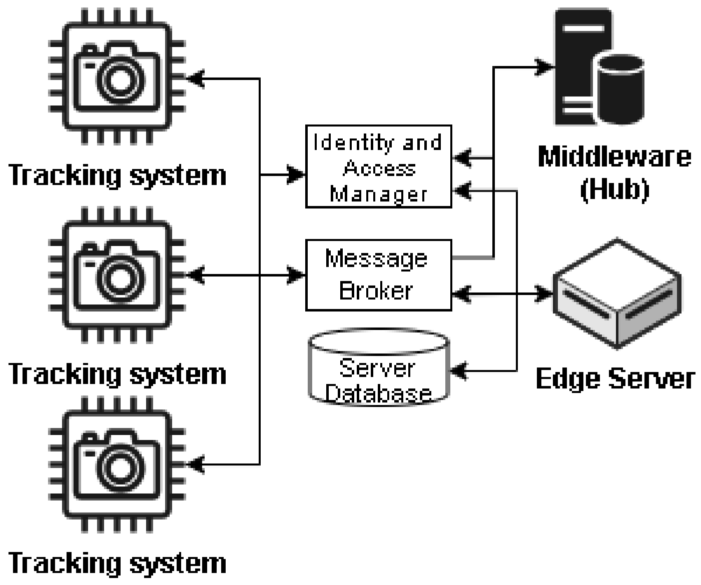


Figure 6: Dissertation architecture.

In the entire project, specific software was selected to implement the required functionalities:

- Home Assistant: This software was selected because of its versatility when using different types of network communication and integration capacity with different sensor classes. This works as a hub for the global system.
- InfluxDB: This database stores information from the tracking systems. It was selected because it implements a temporal database, which supports relevant data search and analysis functions, envisioning an optimised framework to explore data at an application level.
- Mosquitto Broker: This message broker enables communication between different machines. Other software implements the MQ Telemetry Transport(MQTT) protocol, like RabbitMQTT and EMQ X. It was chosen mainly because it was a plugin in the home assistant software.
- Redis: This database is used by the Edge Server to store the configuration data from the tracking system. This database was chosen because of its simplicity and the low amount of data that will be stored. This does not use the query languages of traditional databases.
- Keycloak: This identity and access management software is responsible for authenticating all the clients, the tracking systems and the applications that can be developed in the future. This software was chosen because it is open source, supports different protocols, has well-written documentation, and has a big community.

4 System Implementation

This chapter will provide a detailed description of the techniques, algorithms, and procedures used to create the proposed solution for the people tracking aspect of the Lab4USpaces project. Figure 7 (a rearranged version of Figure 5 that best shows the MOT system components) illustrates the proposed architecture, including the software employed by each project component. This chapter will focus on the tracking systems and edge server components, their interaction with the software, and a comprehensive analysis of the methods, tools, and procedures used for software development and implementation. The chapter will include information about the program design and operation.

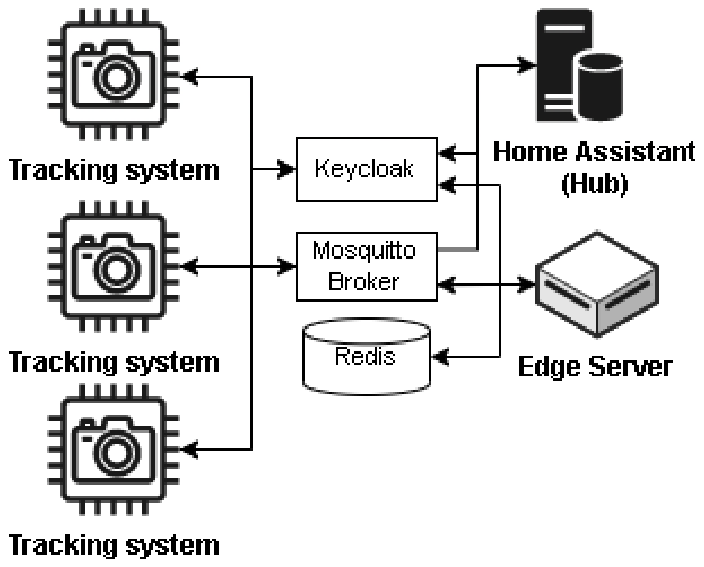


Figure 7: Architecture design with the software.

Global coordinates system

Before diving into the main components of this dissertation it is needed to explain a core concept of the system, that is the global coordinate system which is responsible for creating a unified coordinate system between all the cameras of the tracking systems inside the campus. This system, located in the tracking system, coordinates the positions of cameras and converts their data to the global coordinate system before sending it to the hub. This process involves mapping each camera's field of view (represented by Cam_{width} and Cam_{height}) to the global system and determining the new scale (represented by $ScaleIn_X$ and $ScaleIn_Y$), angle (represented by θ), and offset (represented by $Offset_X$ and $Offset_Y$), which are used to transform the data from each camera. Some more details are presented in chapter 5.1 later in this dissertation.

Equations 4.1, 4.2, and 4.3 are applied when transforming a video camera coordinate to the global coordinate system. $Coord_X$ and $Coord_Y$ correspond to the coordinates of the tracked person within a video camera frame; $ScaleIn_X$ and $ScaleIn_Y$ is a scale value necessary to adjust a camera's field of view to the correspondent Campus map section; Cam_{width} and Cam_{height} are the original dimensions of the frame; these variables are used to determine the $Scale_X$ and $Scale_Y$ which are then used to get the $Rotation_X$ and $Rotation_Y$ using the cosine and sine of the rotation angle θ. To get the final coordinates $Final_X$ and $Final_Y$ the new rotated coordinates are summed to the $Offset_X$ and $Offset_Y$ values which represent the values to get the newly scaled image to the correct position.

$$Scale_X = \frac{(Coord_X * ScaleIn_X)}{Cam_{width}} \qquad Scale_Y = \frac{(Coord_Y * ScaleIn_Y)}{Cam_{height}} \tag{4.1}$$

$$Rotation_X = Scale_X * \cos\theta - Scale_Y * \sin\theta \qquad Rotation_Y = Scale_X * \sin\theta + Scale_Y * \cos\theta \tag{4.2}$$

$$Final_X = Rotation_X + Offset_X \qquad Final_Y = Rotation_Y + Offset_Y \tag{4.3}$$

4.1 Technologies Used

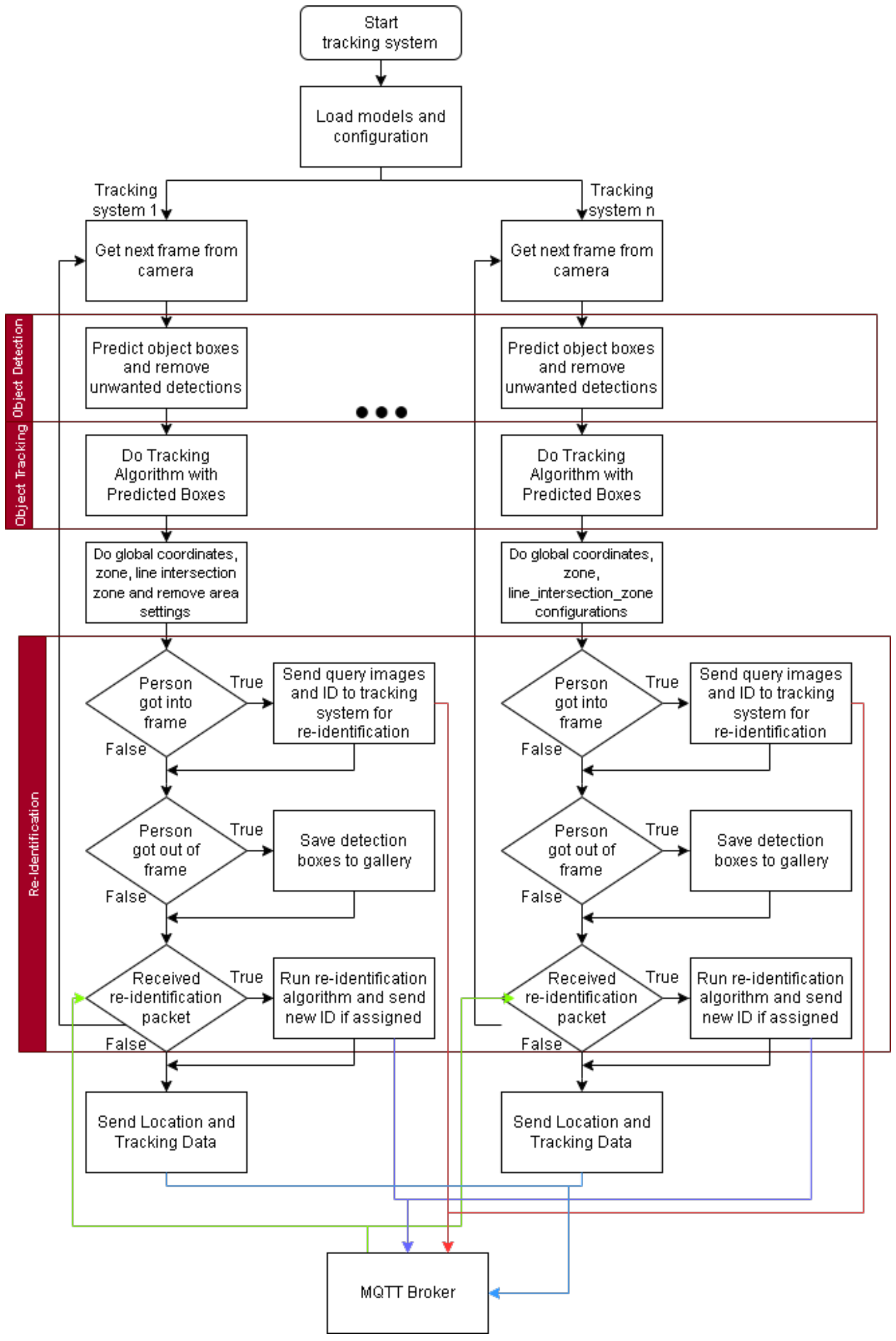
The research presented in this dissertation utilises several technologies to achieve the objectives. The main technologies employed include Python, Flask, Docker, Git, and Visual Studio Code. These technologies were chosen for their versatility and ability to support developing and deploying a highly scalable and efficient solution.

Python was chosen as the programming language due to its ease of use and wide range of available libraries and modules, particularly in machine learning and computer vision. Flask was selected as the web framework to support the development of a RESTful API for communication with the tracking system.

Docker was used for packaging and deploying the tracking system and the edge configuration server, allowing easy and consistent deployment across different environments. Git was employed for version control and collaboration, enabling multiple team members to work on the project simultaneously. Finally, Visual Studio Code was used as the integrated development environment (IDE), offering various features and tools to support efficient and effective coding.

4.2 Tracking System

The core aspect of this dissertation project is a multi-tracking system responsible for collecting, analysing, and transmitting data within the global system. It employs object detection, tracking, and re-identification methods on a live camera feed and then sends the processed data to the central hub. The system was built using Python programming language. The basic data flow structure of the tracking system is illustrated in Figure 8, which presents two tracking system instancies, each one has an associated video camera for live streaming, and they are interconnected, with all other correspondent blocks, through the MQTT broker used to deliver inter-process communication messages. The system is designed to allow full scalability, limited only by the resources available. Each component will be explained in more detail in the following paragraphs.



Start tracking system
Load models and configuration
Tracking system 1
Tracking system n
Get next frame from camera
Object Detection
Object Tracking
Predict object boxes and remove unwanted detections
Do Tracking Algorithm with Predicted Boxes
Do global coordinates, zone, line intersection zone and remove area settings
Do global coordinates, zone, line_intersection_zone configurations
Re-Identification
Person got into frame
True
False
Send query images and ID to tracking system for re-identification
Person got out of frame
Save detection boxes to gallery
Received re-identification packet
Run re-identification algorithm and send new ID if assigned
Send Location and Tracking Data
MQTT Broker

Figure 8: Basic system flowchart

To commence the tracking system, the program must first authenticate itself with the Keycloak framework (a requirement imposed by a global system architecture design decision), using the arguments passed through the command line. Afterwards, the configuration file is opened and interpreted. The configuration parameters are listed and briefly explained in Table 4.

Table 4 Configurations file description

Configuration Name	**Description**
ip	IP of the MQTT Broker
camera_id	ID number of the tracking system
camera_name	Name of the tracking system
camera_zone	Name corresponding to the zone of camera capture
timestamp_config_creation	Date and time of the last config change
weights	Object detection model used
source	Source of the video camera
iou_thres	Threshold for IoU and NMS
conf_thres	Object confidence threshold
img_size	Inference image size
cmc_method	Camera motion compensation method
track_high_thresh	Tracking confidence threshold
track_low_thresh	Lowest tracking threshold
new_track_thresh	New tracking threshold
aspect_ratio_thresh	Threshold for filtering boxes with unaccepted aspect ratio
classes	List of detection classes
folder_remove_seconds	Maximum amount of seconds that a person can be re-identified after they were last seen
ReID_mean_threshold ReID_median_threshold ReID_mode_threshold	Thresholds for the re-identification to occur with the mean, median and mode values
global_map_scale global_map_angle global_map_offset	Values corresponding to the new representation of the frame in the unified coordinate system, with scale, angle and offset values
zone	Zone configuration contains the name of the zone inside and outside and a list of points
line_intersection_zone	Line intersection zone configuration contains a name, start and end points, the name for each crossed direction as well as an ID association field which contains the publishing location and the name of the associated config
remove_area	Remove area configuration, which contains a list of points that construct a polygon

The program starts by creating a thread to send the configuration data to the hub so that the administrator can monitor the system's status and adjust the settings. The thread then continues to listen for incoming messages.

After loading the live stream camera and setting up the detection model and tracking algorithms, the program extracts prediction boxes from the current frame. Unwanted boxes are eliminated by applying non-maximum suppression (NMS), and the remaining boxes are fed into the BoT-Sort tracking algorithm. This algorithm employs techniques to predict the final output tracks and assign IDs to each prediction. The final values are then stored and processed in a class where some validations happen, like removing people that have been undetected for a long time and associating the other objects with the corresponding person. This class is also where the main configuration settings are used, the "zone" feature determines the number of people in a specific area, while the "line_intersection_zone" is used for re-identification and counting the number of people crossing it.

If an ID association exists in the "line_intersection_zone" configuration (see also table 4) re-identification can occur. The process begins by detecting when a person crosses the configuration line and using the sequence of centroids (the centre point of the detection boxes) to determine the person's direction (entering or leaving the camera's field of view). If a person is leaving, the same person will likely be found in the associated tracking system, so the last twenty detection boxes of the person are stored on the device, in a specific location, for future re-identification, this number was selected after experimenting and noticing that above twenty images there was no evident precision gain. If a person is entering, it is possible that they came from the associated system, so a packet is created with the current ID and the detection boxes of the person and sent to the corresponding tracking system for re-identification.

Finally, a packet is created with the gathered information before the next frame is processed. It is translated into the unified coordinate system to be able to track everyone throughout the campus, and only then is it sent to the hub through the message broker.

The thread running in parallel with the main tracking process, as said before, is responsible for handling incoming data from other devices. The following paragraphs outline the types of messages that can be received:

- **Update**: This message is solely received from the edge configuration server and contains a new configuration. The settings are updated in the running program and written to the configuration file.
- **Refresh**: This message is received only from the edge configuration server. Its purpose is to verify if the tracking system is running by sending an acknowledgement with the configuration settings.
- **Re-identification**: This message is received from other tracking systems and aims to associate the person boxes received in the packet with a corresponding person in the gallery folder – the retention policy is discussed below. Before re-identification, the saved gallery images of each person are verified for how long they have been saved; if they pass through a specific configured time interval, the photos are deleted, meaning that the person in the received message either was not identified in the tracking system or didn't pass through the field of view. The images in the packet also referred to as query images, are stored locally on the device. The directory of all the gallery and query images is used to create a new dataset that is used in the re-identification algorithm, using the osnet_ain_x0_25 model, the features are extracted, and the cosine metric distance is determined after getting the similarity values between the gallery and query images a mean, median and mode values for each gallery ID is extracted, if they highest values are above a certain threshold means that the query person was re-identified. After the re-identification, the query images are deleted despite being re-identified or not. If a person passes the threshold, the gallery images corresponding to the person selected are deleted, as they have already been re-identified. The new ID is sent back to the original system through the message broker.

- **Reid-association**: This message is received from another tracking system after completing the re-identification task. It contains the new ID recovered and changed to the old one.

Image retention policy

To enable the re-identification module to function properly, it is necessary to store images on the device. However, this can potentially compromise the privacy and security of the individuals being tracked by the system. It should be noted that there is no module that allows external sources to access these images. During system run time, specifically the re-identification process, certain images are stored and others are deleted based on whether any re-identification has taken place or if a certain time threshold has been exceeded. The only means of accessing these images would be through unconventional methods, such as hacking the system.

4.3 Edge Configuration Server

The Edge Configuration Server is designed to manage, adjust the settings and synchronise the operation of all the tracking systems. It is built using the Flask Python web framework and web development languages, such as HTML, JavaScript, and CSS. The server utilises Keycloak to authenticate and authorise each device, Redis to store system configurations, and Mosquitto Broker to communicate with the Hub (Home Assistant) and the tracking systems.

The server after setting up all the connections to all the software retrieves data from the Redis database so it can be used during the execution. A separate thread is established for the web server to handle incoming HTTP requests from users and provide the appropriate response. The primary process takes incoming data packets from the tracking systems via the MQTT Broker, Its basic functionality is to store received configuration packets of the tracking systems in the Redis database so that they can be viewed and changed. The web server has been developed with multiple HTTP routes to achieve its intended functions, which will be further detailed as follows:

- **GET /**: This route renders the home page HTML.
- **GET /updateDataServer**: This route retrieves the basic settings of each existing tracking system from the Redis database and sends them in JSON format to be listed on the home page.
- **GET /refresh_all**: This route requests an acknowledgement from each tracking system to check if they are running. Updates each device's status settings to false, waiting for the system to receive the response to turn it back to true.
- **POST /config**: The purpose of this route is to load the configuration page for a specific tracking system. When a POST request is made, it includes the name of the particular system, and the HTML page and settings are sent to the user.
- **POST /config_set**: This route saves any changes made on the Config HTML page. It saves the new information in the Redis database and sends the updated settings to the corresponding tracking system.
- **POST /config_points**: This route displays an HTML page for one of the three main configuration settings of a tracking system, which include ”line_intersection_zone”, ”zone”, and ”remove_area”. On this page, users can add new settings, make changes, or remove existing settings by drawing on the frame of the tracking system.

- **POST /config_points_set**: This route enables the client to save the changes made to one of the three configuration settings in the database.
- **GET /id_association**: This route renders an HTML page that allows for a specific function, re-identification, to be performed on the global system. It uses each tracking system's "line_intersection_zone" configuration, enabling the administrator to establish associations between the cameras' points of view.
- **GET /id_association/get_association**: This route updates the web page with any changes made to the associations while the client is on the /id_association route. It returns the current associations and the "line_intersection_zone" settings that can still be associated in JSON format.
- **POST /id_association/add_association**: This route is used to transmit the newly updated configuration to the appropriate tracking systems and to change the ID association settings in the Redis database.
- **POST /id_association/remove_association**: This is used when an association is deleted, the information of the tracking systems in the Redis database is updated, and the new settings are sent to the corresponding devices.

4.4 Software used

Docker is a tool that makes it easy for developers to build, deploy, and run applications in containers. Containers are lightweight, portable packages that include everything an application needs to run, such as code, run-time libraries, system tools, libraries, and settings. Moving applications between different environments, from development to production, makes moving it simple without worrying about dependencies or configuration issues. Docker uses a containerisation approach, meaning that each container runs as an isolated process and has its own file system, environment variables, and network interfaces. This allows multiple containers to run on the same machine without interfering with each other, making it possible to run various versions of an application or multiple applications with conflicting dependencies on the same host. Docker also provides powerful networking capabilities that enable containers to communicate with each other and the outside world. By default, Docker creates a bridge network for each container, which allows them to communicate with each other through their unique IP addresses. Docker also allows users to create custom networks to isolate and secure containers, preventing unauthorised access from outside the network. Docker provides a range of security features to protect containerised applications. Containers are isolated from the host system and other containers, and Docker uses namespaces and control groups to limit the resources available to each container. Docker also provides support for security best practices, such as using non-root users, running containers with read-only file systems, and limiting access to system resources. These security measures make Docker a reliable and secure tool for building, deploying, and managing containerised applications.

Docker Compose is an additional tool that enhances the functionality of Docker by allowing users to define and run multi-container applications. It provides an easy way to configure and start multiple containers with a single command, simplifying the management of complex applications with various services.

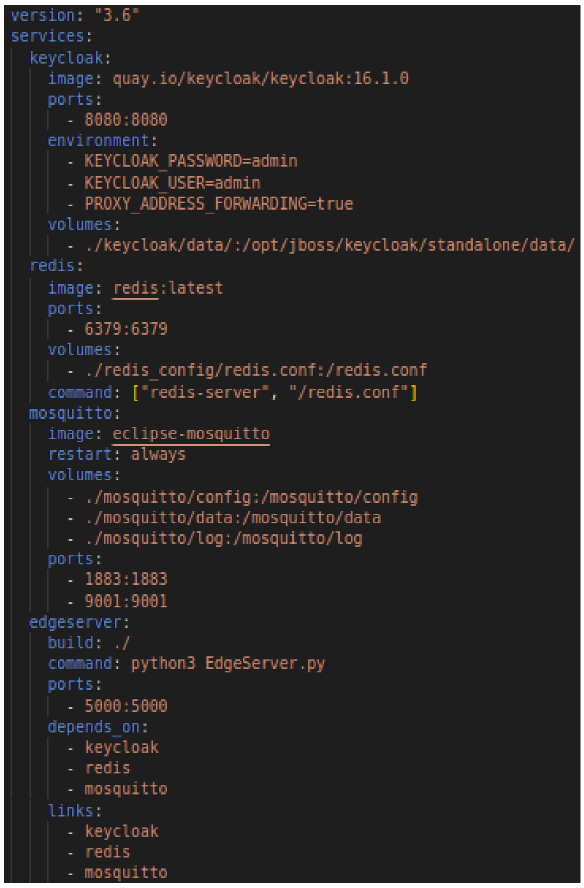
In this project, Docker Compose delivers four different services: Keycloak, Redis, Mosquitto Broker, and Edge server. The Keycloak, Redis and Mosquitto Broker services are pulled from the Docker framework, while the Edge server service uses a custom Dockerfile with all the necessary dependencies. This approach is beneficial as it enables developers to define the specific dependencies and configurations needed for the Edge server service while taking advantage of the convenience and consistency Docker Compose provides for managing the other services.

5 Results and Demonstrations

This chapter provides an overview of the application components, within the Lab4USpaces research project. It begins with a description of the setup phase, during which the docker containers are initialised with the necessary software and configured. This process includes obtaining the crucial global coordinates setting. The chapter will then examine the tracking system in detail, including illustrations of the results, followed by a discussion of the Edge Server web pages. The final section of the chapter will cover the MQTT connections of the entire system.

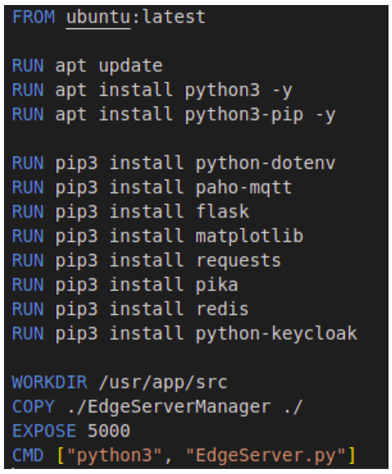
5.1 Setup

Before running the main program, the tracking system modules and all the other software must be set up. The docker-compose yml file, shown in figure 9, makes it easy to launch all the software with just one command. A Dockerfile is also needed to run the Edge Server along with the other software, as depicted in figure 10.



```
version: "3.6"
services:
  keycloak:
    image: quay.io/keycloak/keycloak:16.1.0
    ports:
      - 8080:8080
    environment:
      - KEYCLOAK_PASSWORD=admin
      - KEYCLOAK_USER=admin
      - PROXY_ADDRESS_FORWARDING=true
    volumes:
      - ./keycloak/data/:/opt/jboss/keycloak/standalone/data/
  redis:
    image: redis:latest
    ports:
      - 6379:6379
    volumes:
      - ./redis_config/redis.conf:/redis.conf
    command: ["redis-server", "/redis.conf"]
  mosquitto:
    image: eclipse-mosquitto
    restart: always
    volumes:
      - ./mosquitto/config:/mosquitto/config
      - ./mosquitto/data:/mosquitto/data
      - ./mosquitto/log:/mosquitto/log
    ports:
      - 1883:1883
      - 9001:9001
  edgeserver:
    build: ./
    command: python3 EdgeServer.py
    ports:
      - 5000:5000
    depends_on:
      - keycloak
      - redis
      - mosquitto
    links:
      - keycloak
      - redis
      - mosquitto
```

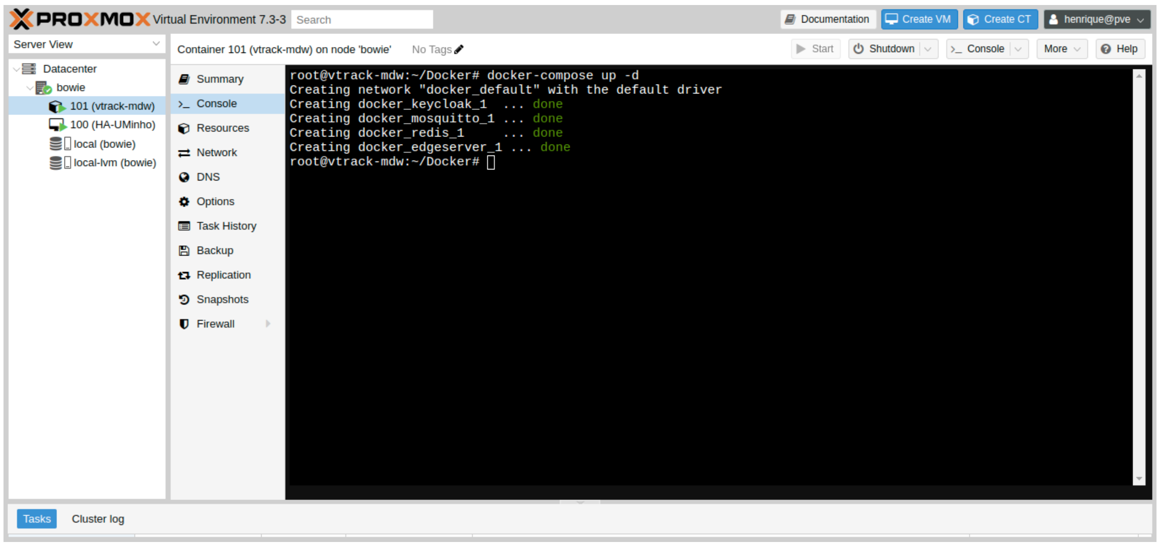
Figure 9: Docker-compose file.



```
FROM ubuntu:latest

RUN apt update
RUN apt install python3 -y
RUN apt install python3-pip -y

RUN pip3 install python-dotenv
RUN pip3 install paho-mqtt
RUN pip3 install flask
RUN pip3 install matplotlib
RUN pip3 install requests
RUN pip3 install pika
RUN pip3 install redis
RUN pip3 install python-keycloak

WORKDIR /usr/app/src
COPY ./EdgeServerManager ./
EXPOSE 5000
CMD ["python3", "EdgeServer.py"]
```

Figure 10: Edge Server docker file.

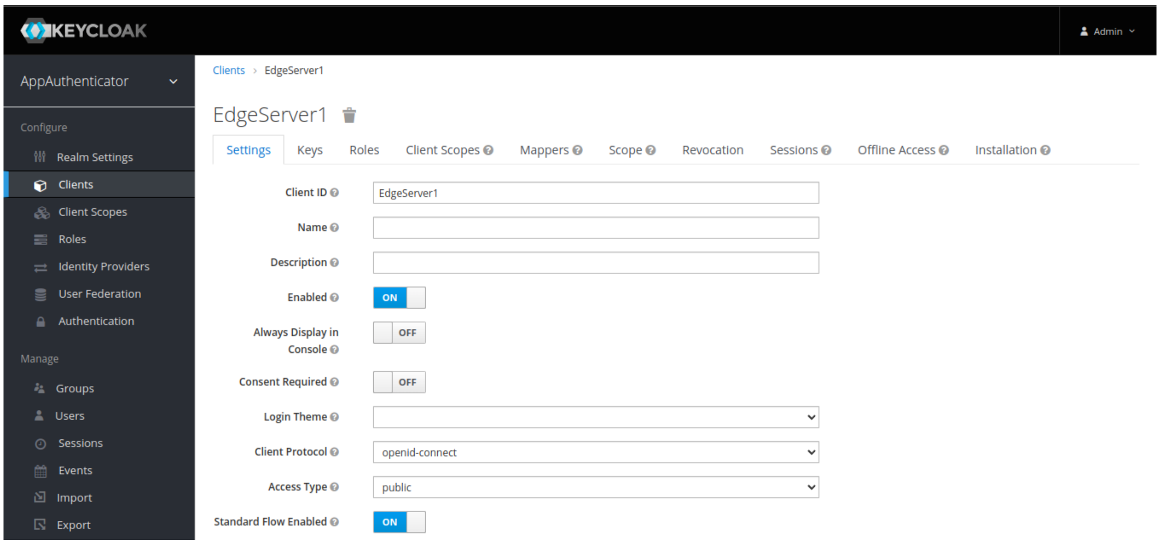
The complete software platform runs on a Proxmox virtual environment, on a local server, and all required files are imported to the system container ("vtrack_msw" in the figure 11). The docker-compose command is executed in order to start all the services as can be seen in Figure 11. Additionally, in this figure can also be seen that a virtual machine is running the Home Assistant (Hub) represented as "HA-UMinho".



```
root@vtrack-mdw:~/Docker# docker-compose up -d
Creating network "docker_default" with the default driver
Creating docker_keycloak_1  ... done
Creating docker_mosquitto_1 ... done
Creating docker_redis_1     ... done
Creating docker_edgeserver_1 ... done
root@vtrack-mdw:~/Docker#
```

Figure 11: Using the container in the local server.

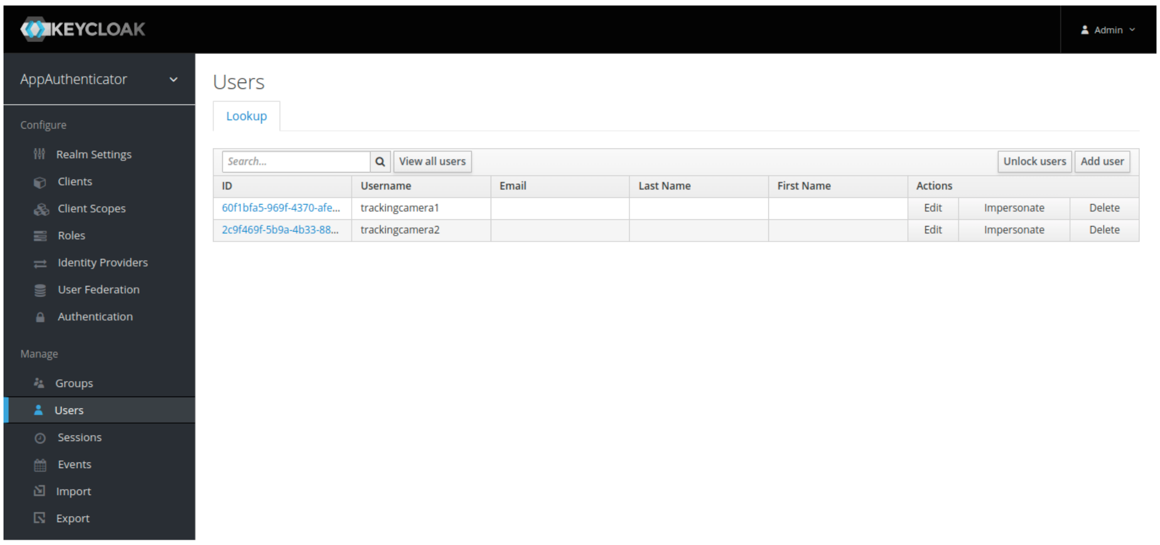
Once the software setup is complete, some configurations in Keycloak need to be made. First, create a new realm called "AppAuthenticator" to manage all the project devices. Then, as shown in Figure 12, create a client for the Edge Server to verify that all devices have the proper tokens. Lastly, create all tracking devices (which are identified in Keycloak by users) so they can request authentication tokens and be identified, as shown in Figure 13.



EdgeServer1

Client ID	EdgeServer1
Name	
Description	
Enabled	ON
Always Display in Console	OFF
Consent Required	OFF
Login Theme	
Client Protocol	openid-connect
Access Type	public
Standard Flow Enabled	ON

Figure 12: Creation of the app client.



Users

ID	Username	Email	Last Name	First Name	Actions		
60f1bfa5-969f-4370-afe...	trackingcamera1				Edit	Impersonate	Delete
2c9f469f-5b9a-4b33-88...	trackingcamera2				Edit	Impersonate	Delete

Figure 13: Users present in the realm.

To obtain the global coordinate information for each tracking device, the scaling information needs to be extracted from a background image of the campus, as can be seen in figure 14).

Figure 15 shows the relevant information for the global coordinates. The dimensions of the campus image are fixed at 2387 width and 1836 height pixels. The extracted information includes the x and y offset values, the re-scaled percentage of the frame, and the angle. The original image dimensions of the frame in the figure are 1280 width and 720 height pixels, but after re-scaling to the global coordinates, the image would only have 36 width and 20 height, resulting in 720 total pixels, which is a low amount. To counter this, the global system was up-scaled by a factor of 50, providing a sufficient number of pixels to be displayed in the applications.

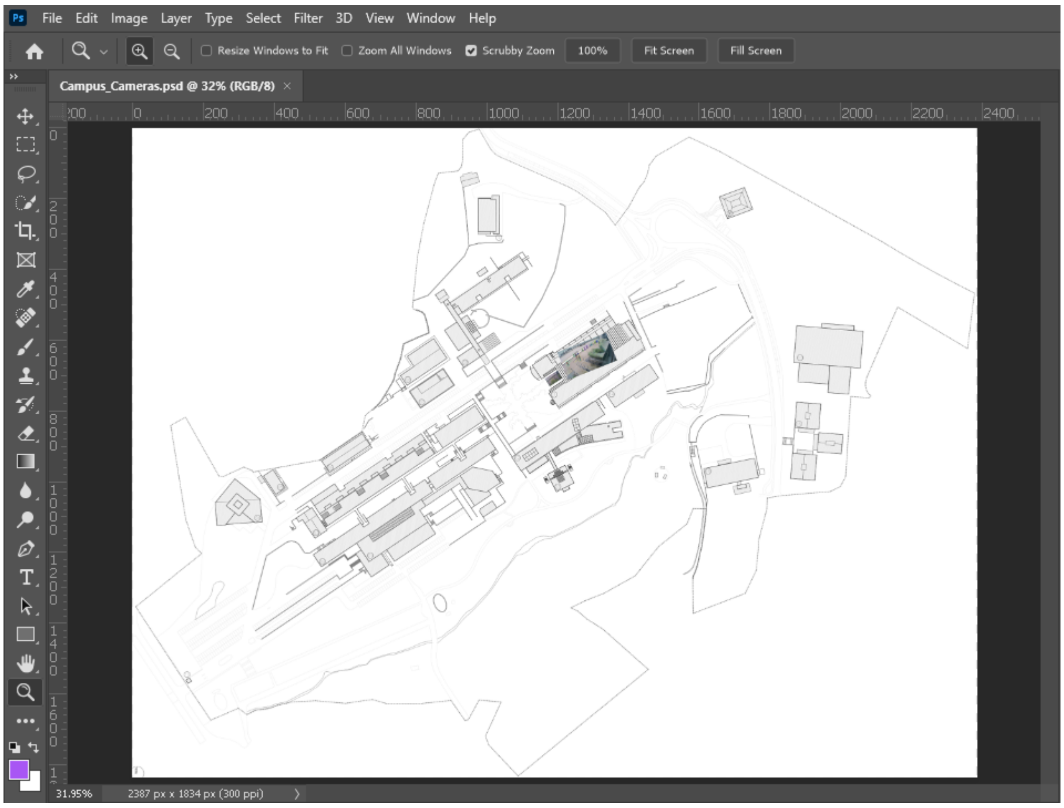
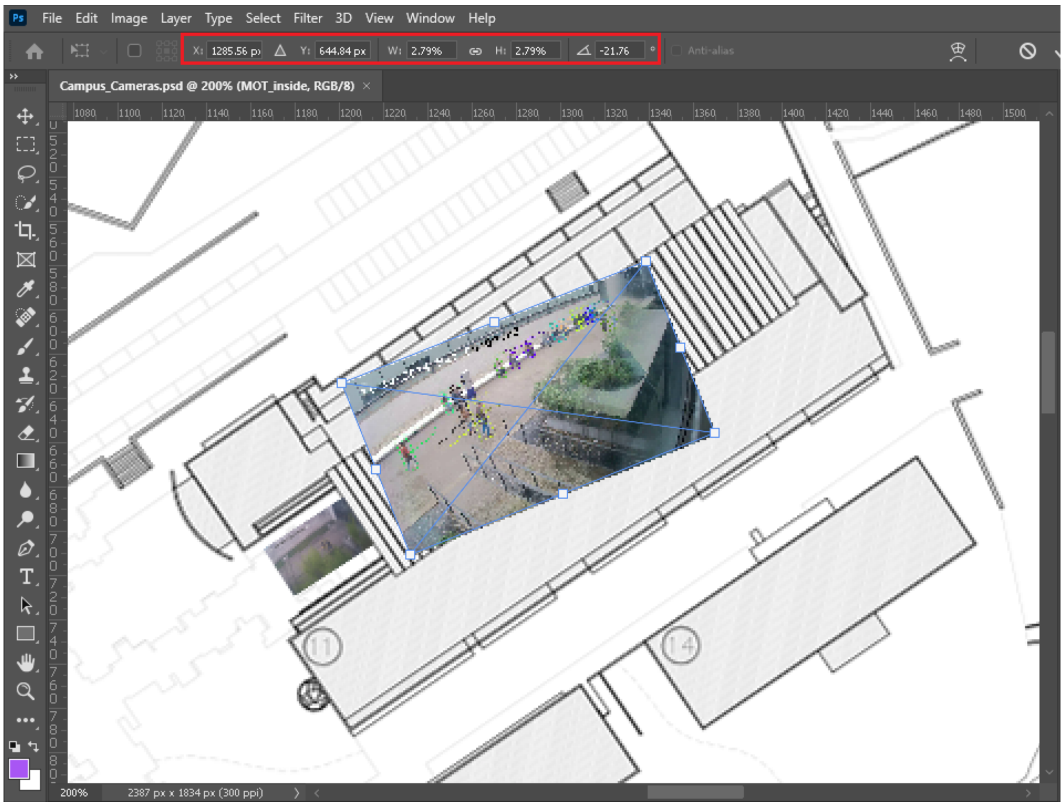

File Edit Image Layer Type Select Filter 3D View Window Help
Resize Windows to Fit
Zoom All Windows
Scrubby Zoom
100%
Fit Screen
Fill Screen
Campus_Cameras.psd @ 32% (RGB/8)
31.95%
2387 px x 1834 px (300 ppi)

Figure 14: Campus global coordinate system.


File Edit Image Layer Type Select Filter 3D View Window Help
Anti-alias
Campus_Cameras.psd @ 200% (MOT_inside, RGB/8)
200%
2387 px x 1834 px (300 ppi)

X: 1285.56 px Y: 644.84 px W: 2.79% H: 2.79% ∠ -21.76 °

Figure 15: Extracted re-scaled information.

5.2 Tracking System

In order to demonstrate the tracking system's capabilities, two video cameras were installed on campus. The tests were conducted using two videos captured by those cameras, one inside a building and the other outside, without any overlap. Both videos had an original resolution of 1280x720. However, for performance reasons and without compromising the detection capacity, the resolution of both videos was scaled-down to 640 x 360. Both videos comprise one minute in length. The testing was performed on two different computers, one with an Intel Core i7-8550U CPU 1.80GHz with 8GB of RAM, and the other with an AMD Ryzen 7 5700U with 16GB of RAM.

The current testing environment is not optimal, as it would be preferable to use live cameras instead of prerecorded videos. The processing time between frames is always different, but when using prerecorded videos as the system always processes the next frame it creates a time mismatch between the videos, and this is even more highlighted when there is a big difference in performance between the devices running the program. When using live cameras this would never happen because the system always extracts the most recent frame.

The following commands were executed on each device to run the tracking system and set up the corresponding configuration files. The argument "view-img" was used to view the device activity, but this should not be used in the final deployment of the tracking system to ensure the privacy and security of the tracked individuals.

```
    python3 tracking_system.py --username trackingcamera1 --
password trackingcamera1 --config ./config/config.json --view-img --device cpu
```

```
    python3 tracking_system.py --username trackingcamera2 --
password trackingcamera2 --config ./config/config2.json --view-img --device cpu
```

Figures 16 and 17 show the two main methods being used: object detection and multiple object tracking, using the YOLOv7 and the BoT-SORT models, respectively. The time registered to detect and track the objects is between 142.8ms (7 FPS) and 263ms (3.8 FPS). These values are within the recommended frame rate for this type of application in order not to lose operation capability, as stated in (Mohan et al., 2018) – the authors indicate a good working point of 6 FPS with 0% accuracy loss, being acceptable to reduce the FPS by 80% (1.2 FPS) and keeping precision rates above 60%.

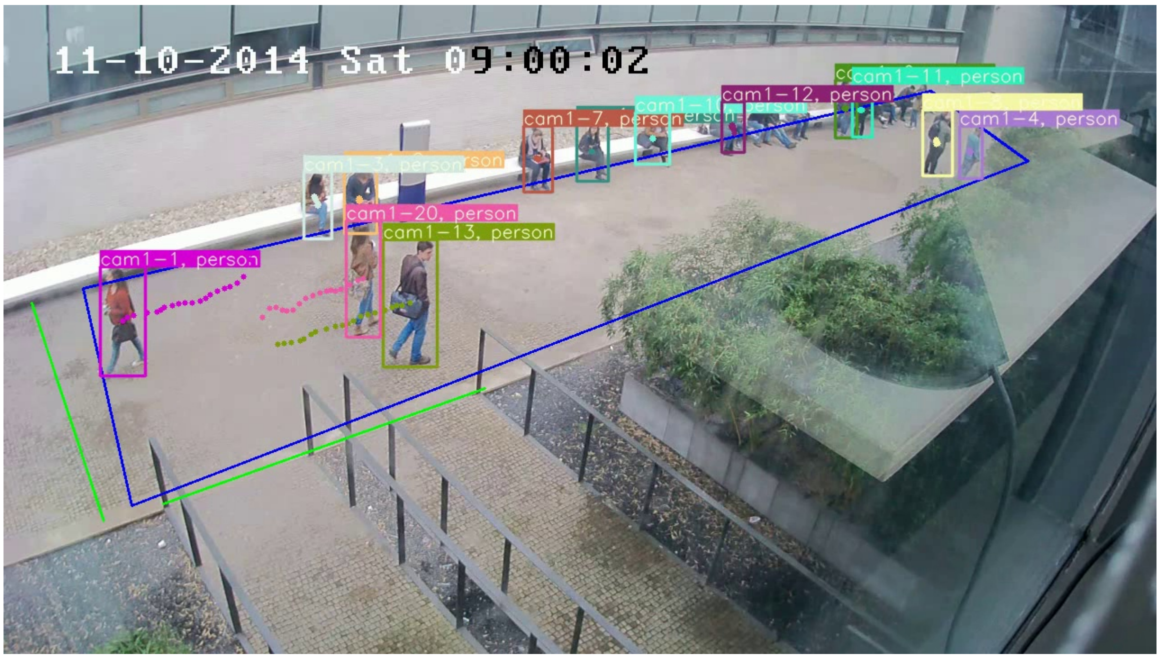


Figure 16: Object detection and multiple object tracking inside the building.

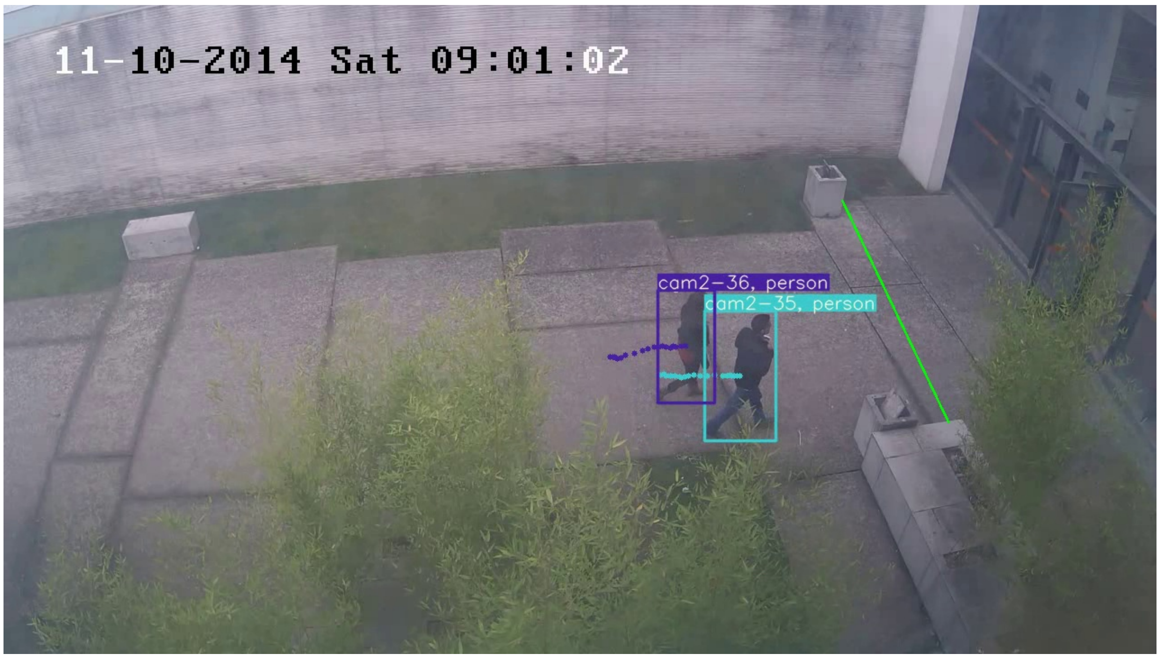


Figure 17: Object detection and multiple object tracking outside the building.

To demonstrate the final method of the tracking system, the re-identification, using the Deep Person Reid algorithm with the model osnet_ain_x0_25, as indicated previously (see section 3.1.3), three figures will be explained and compared. The tracking systems communicate with each other to check if the associated camera has already detected the person that entered a scene. This checking occurs whenever a new object is detected entering a predefined zone. Likewise, each camera will record the object's identification whenever it leaves the scene by predefined zones. Figure 18 shows a person exiting the outdoor scene with the label "cam2_1" (indicating that camera two is the detecting device and that the person ID is 1). Figure 19 shows the same person entering the indoor camera's field of view a few seconds later and being labelled "cam1_22" (which indicates that camera one is the detecting device and that the person ID is 22), this person has not yet been re-identified. Finally, in figure 20, it is visible that the ID assigned by camera two has been changed to the one previously set by camera one.

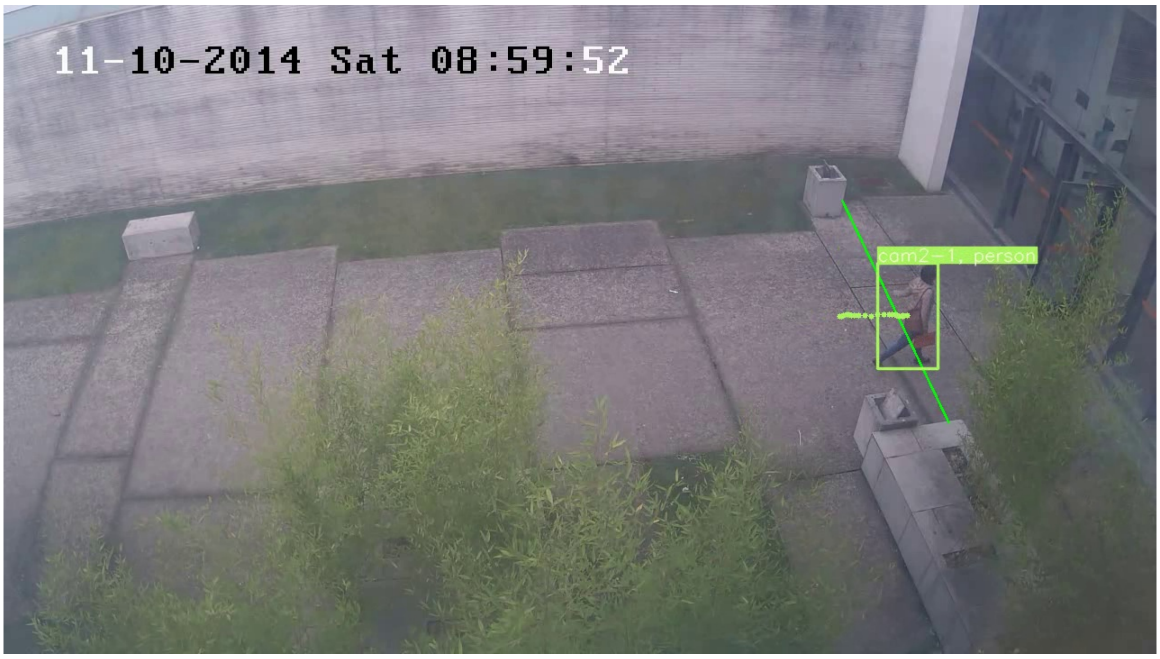


Figure 18: A person leaving the outdoor scene.

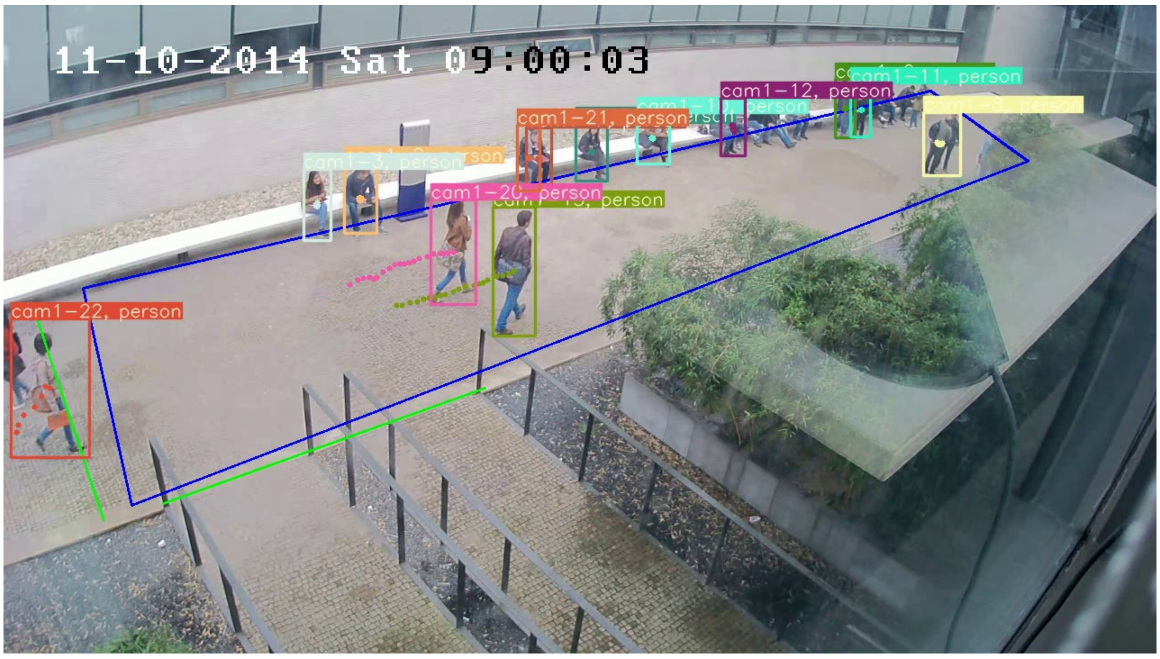


Figure 19: Another tracking system is detecting the person.

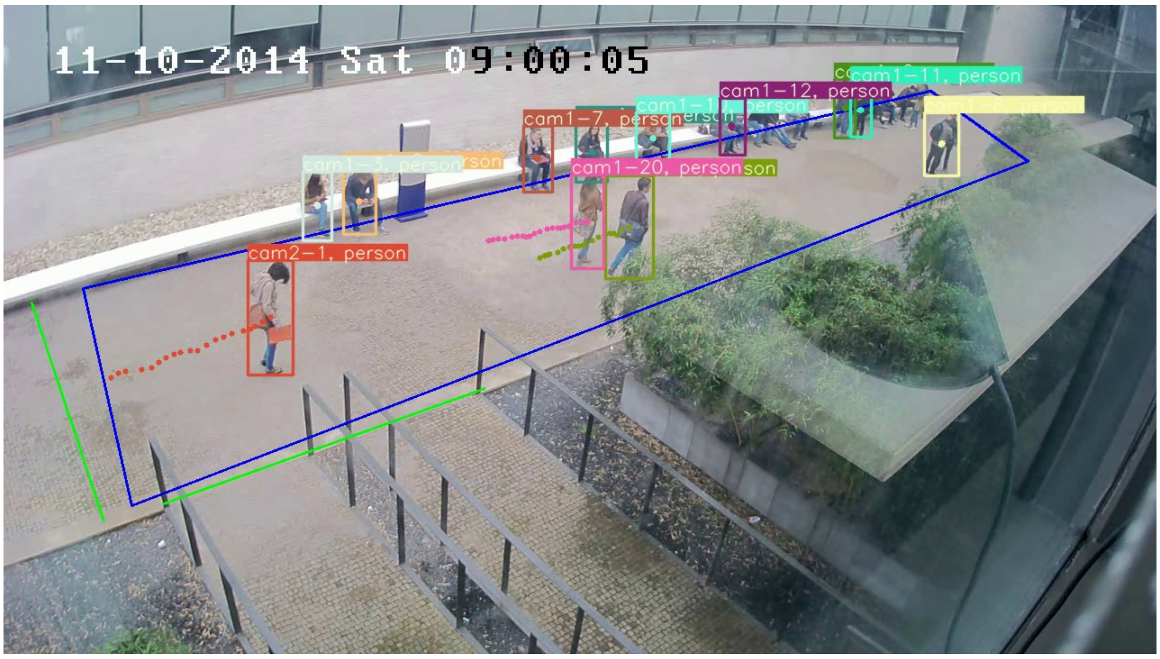


Figure 20: Re-identification of the person from another camera.

To synthesise the results from the tests, they were categorised into detection, tracking and re-identification results. Detection results were measured by the number of accurately identified objects, with a maximum of 25 people being observed, of which 17 were correctly identified in this instance. The most frequent errors in detection were occlusions and low resolution when individuals moved away from the camera. The tracking results were evaluated based on the percentage of ground-truth tracks that had the same label for at least 80% of their lifespan, referred to as Mostly Tracked (MT). In the small dataset that was tested, the MT result ranged between 50% and 70%. Regarding re-identification, the small dataset achieved three correct identifications out of a possible four, indicating a 75% efficacy rate. However, further experiments are necessary to validate this finding.

All the detection and tracking information is processed and sent to a hub for analysis, management and storage. This data and the resulting analysis can be used in several applications, such as the one shown in Figures 21 and 22, which displays the concentration of people in the monitored campus spaces using colour and bubble size codes. Figure 21 shows the full campus map, while Figure 22 shows the result of zooming in on a particular corridor.

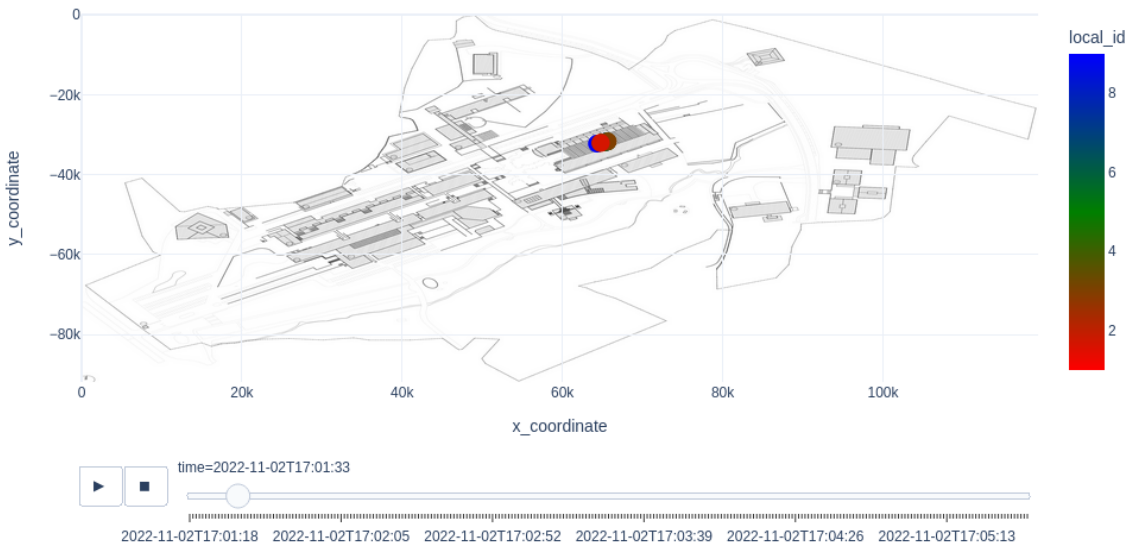


Figure 21: Example application showing the tracking points on the campus.

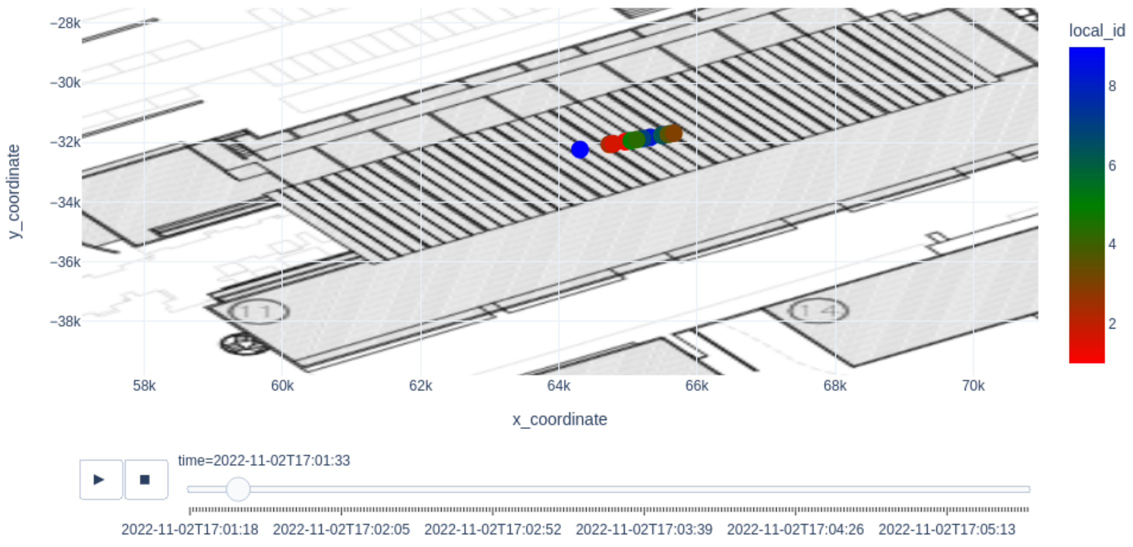
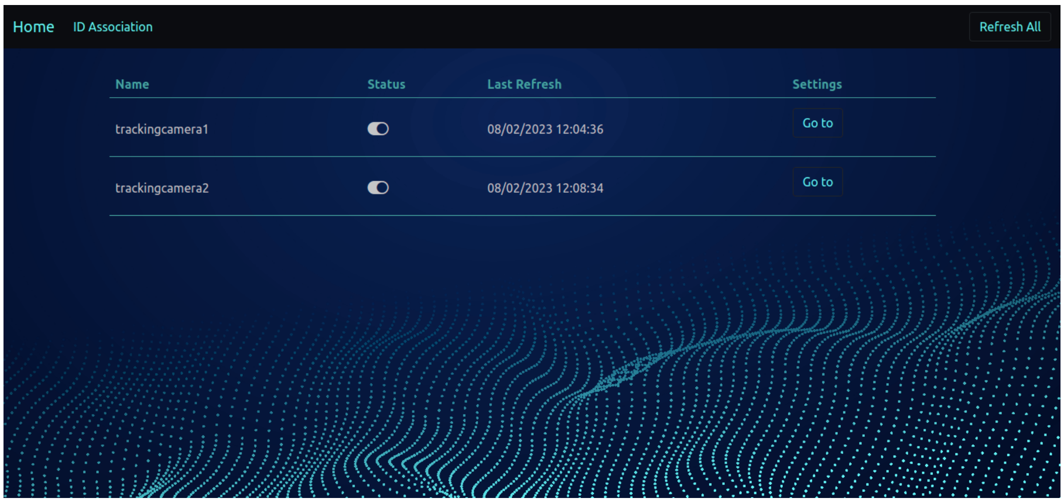


Figure 22: Example application showing the tracking points on the campus zoomed in.

5.3 Edge Configuration Server

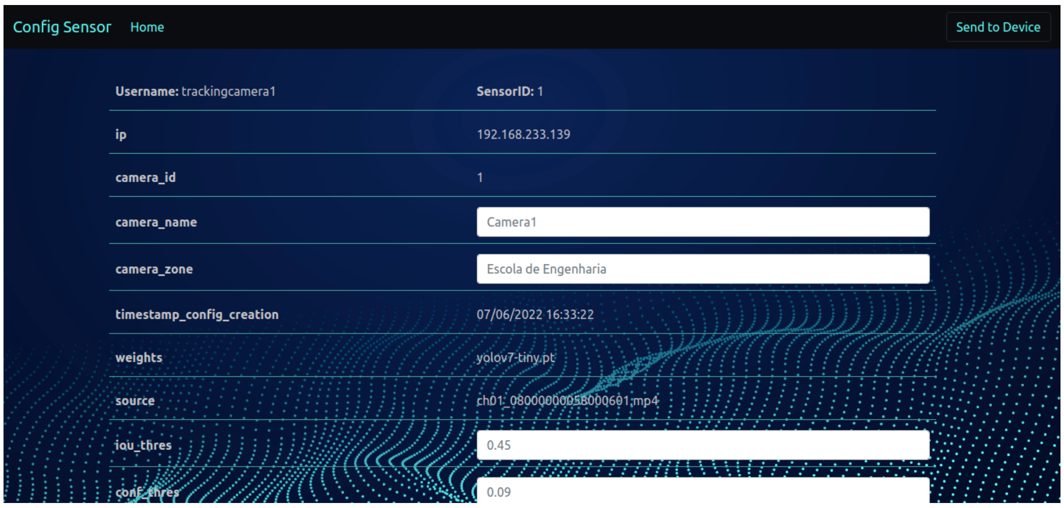
This section presents a description of all the different pages used by the Edge Configuration Server. Firstly, figure 23 shows the home page of the configuration server, where all the tracking devices can be seen, as well as their status, and the last time they were refreshed.



Name	Status	Last Refresh	Settings
trackingcamera1		08/02/2023 12:04:36	Go to
trackingcamera2		08/02/2023 12:08:34	Go to

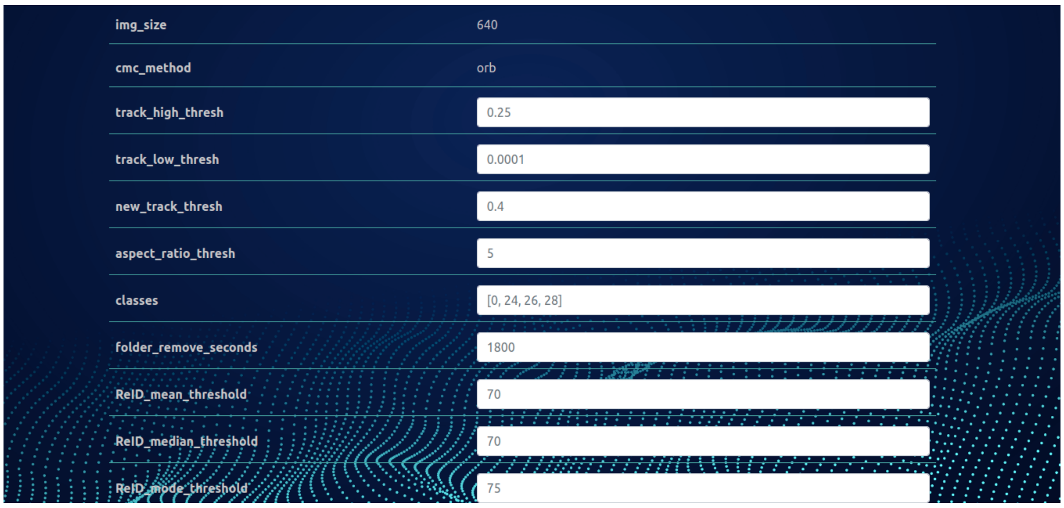
Figure 23: Home page of the Edge Server.

Figure 24, 25 and 26 shows the settings related to a tracking system that is running. Here, the configurations can be viewed and changed in order to modify the results obtained in the running system, this information viewed is explained in table 4. At the end of the page, three buttons can be seen, they are used to route to the three main configurations.



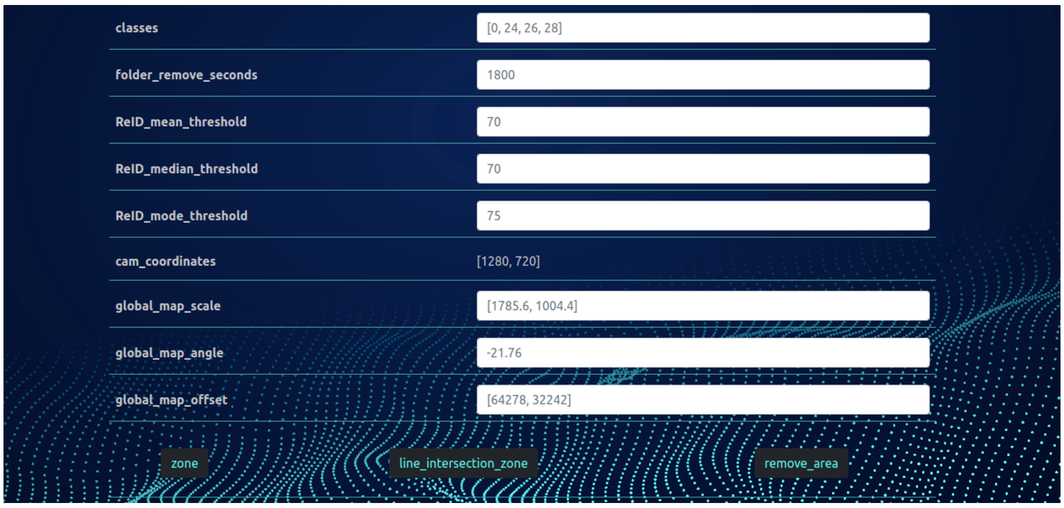
Username: trackingcamera1	**SensorID:** 1
ip	192.168.233.139
camera_id	1
camera_name	Camera1
camera_zone	Escola de Engenharia
timestamp_config_creation	07/06/2022 16:33:22
weights	yolov7-tiny.pt
source	ch01_08000000058000601.mp4
iou_thres	0.45
conf_thres	0.09

Figure 24: Configuration page of a tracking system part 1.



img_size	640
cmc_method	orb
track_high_thresh	0.25
track_low_thresh	0.0001
new_track_thresh	0.4
aspect_ratio_thresh	5
classes	[0, 24, 26, 28]
folder_remove_seconds	1800
ReID_mean_threshold	70
ReID_median_threshold	70
ReID_mode_threshold	75

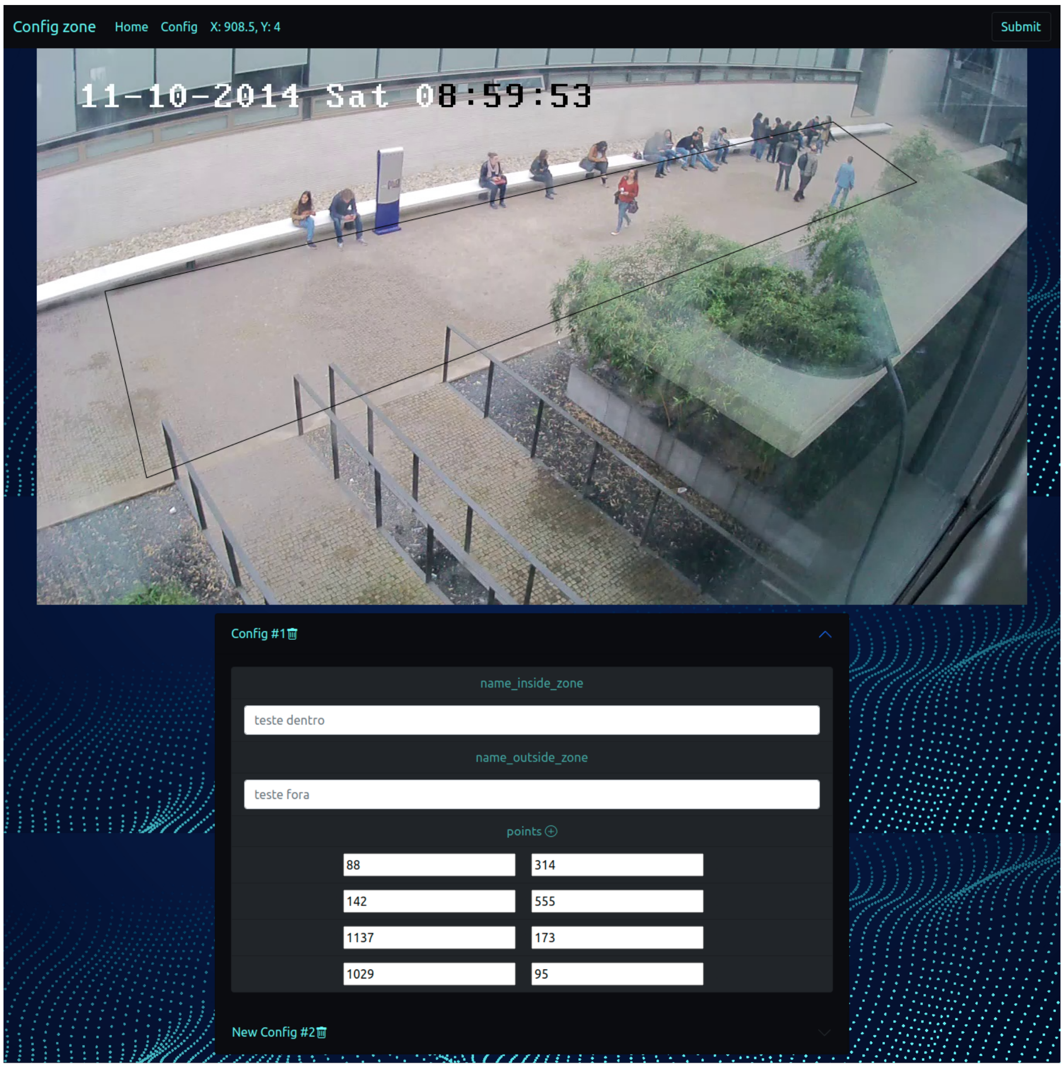
Figure 25: Home page of the Edge Server part 2.



classes	[0, 24, 26, 28]
folder_remove_seconds	1800
ReID_mean_threshold	70
ReID_median_threshold	70
ReID_mode_threshold	75
cam_coordinates	[1280, 720]
global_map_scale	[1785.6, 1004.4]
global_map_angle	-21.76
global_map_offset	[64278, 32242]

Figure 26: Home page of the Edge Server part 3.

In Figure 27, the zone configuration is displayed, allowing the administrator to edit or delete existing settings and create new ones by clicking on the frame presented on the page. When defining each point, a polygon is drawn on the image, and each point can be modified to specific coordinates. Moreover, the administrator can provide additional information such as the name of the created zone, both inside and outside.



name_inside_zone: teste dentro

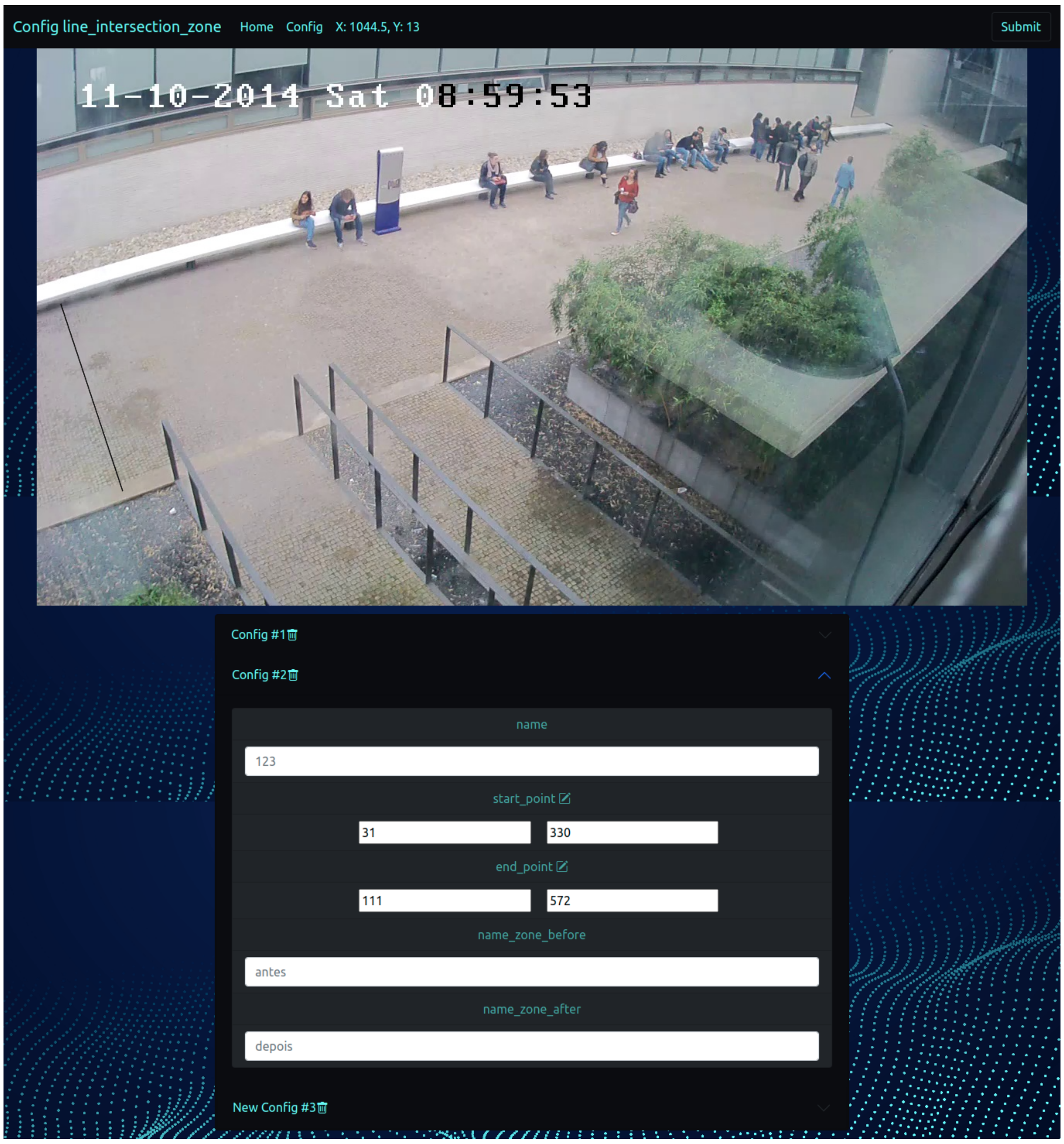
name_outside_zone: teste fora

points

88	314
142	555
1137	173
1029	95

Figure 27: Zone configuration settings.

Figure 28 illustrates the Line Intersection Zone configuration, enabling the administrator to add, edit, and remove configurations. Since this zone is defined by a line, only two points can be set. This configuration is crucial for re-identification, making it recommended to place the lines close to previously known intersection zones. However, this is not always necessary in all settings. Furthermore, a configuration name must be provided, along with the names of the zones before and after the line.



name: 123

start_point: 31 330

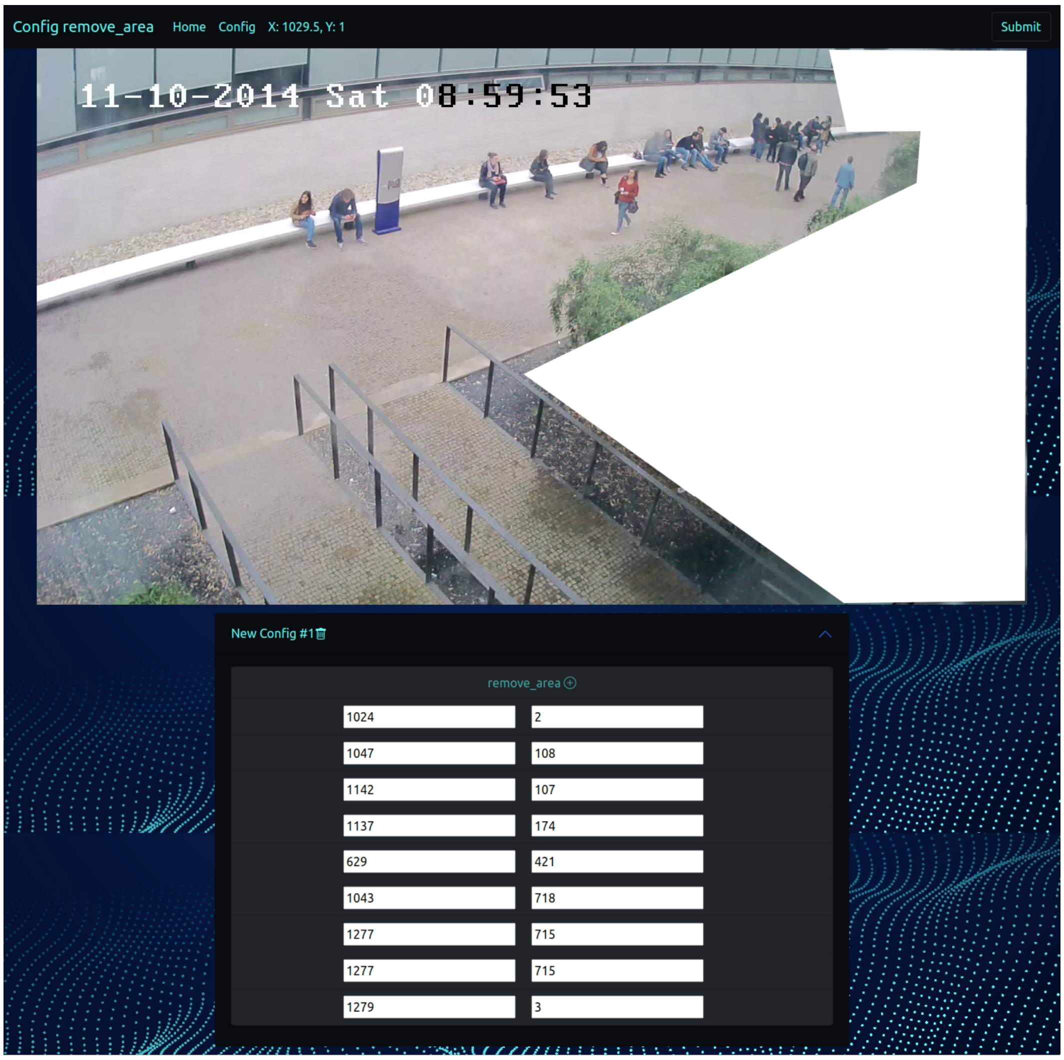
end_point: 111 572

name_zone_before: antes

name_zone_after: depois

Figure 28: Line intersection zone configuration settings.

In figure 29 is displayed the Remove Area configuration which enables the removal of unwanted areas from the frame by drawing a polygon on the image. Like the other configurations, settings can be added, edited, and deleted.



New Config #1

remove_area

1024	2
1047	108
1142	107
1137	174
629	421
1043	718
1277	715
1277	715
1279	3

Figure 29: Remove area configuration settings.

In the ID association page are displayed all the existing associations between two line intersection configurations as well as being able to create new ones by selecting the tracking system and then the line intersection settings as can be seen in figures 30 and 31. This setting is required for the re-identification to occur between two tracking systems.

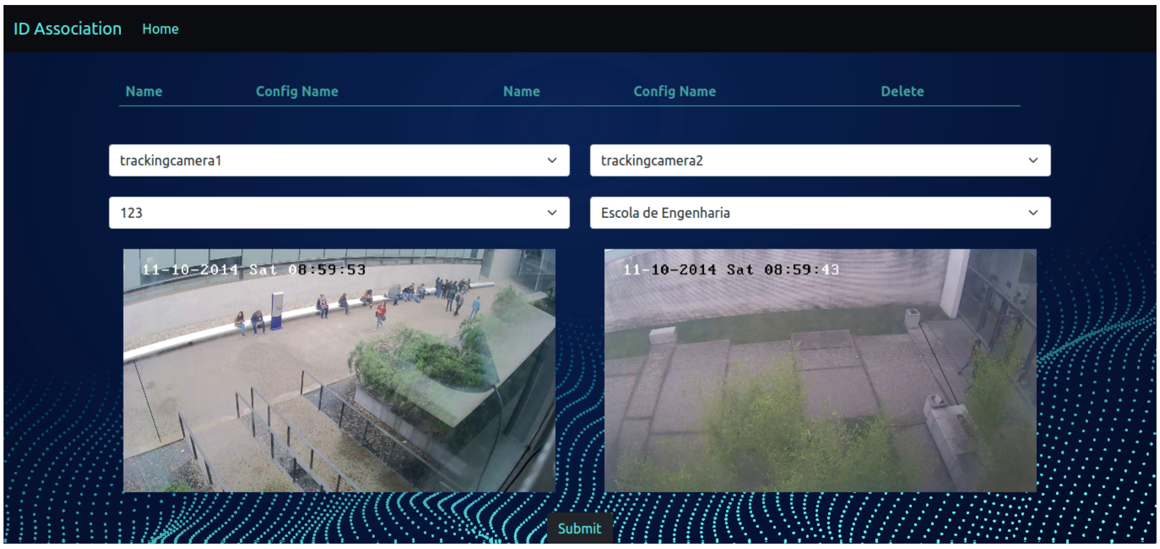
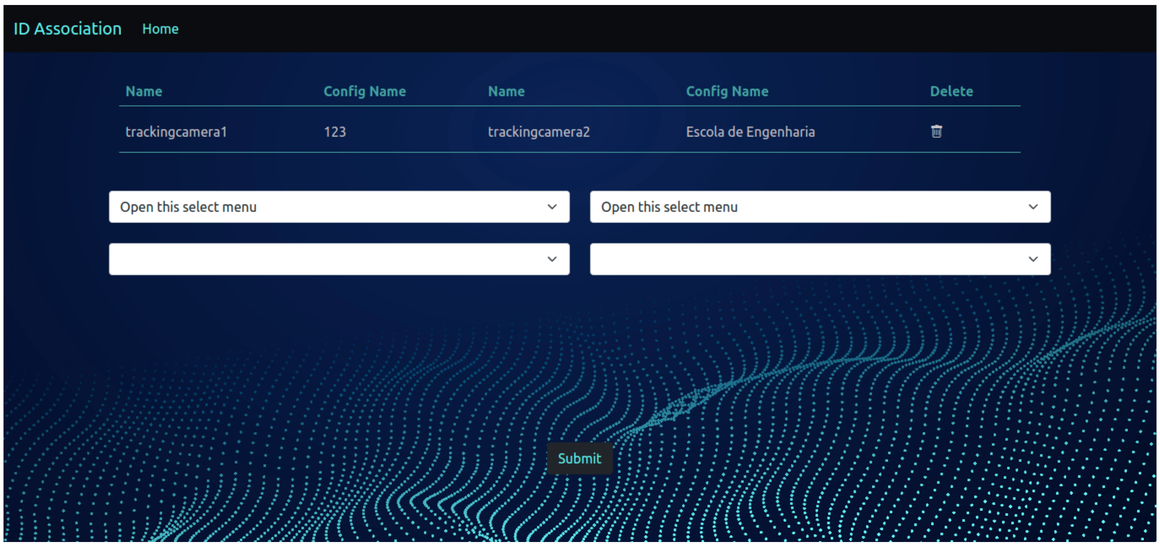


Figure 30: Example application showing the tracking points on the campus.



Name	Config Name	Name	Config Name	Delete
trackingcamera1	123	trackingcamera2	Escola de Engenharia	🗑

Figure 31: Example application showing the tracking points on the campus.

5.4 Protocols Summary

In this section, the objective is to describe all the network interactions that occur during the operation of the entire system. Firstly, a brief overview of the MQTT broker should be provided. MQTT (Message Queuing Telemetry Transport) is a publish-subscribe protocol for Internet of Things (IoT) communication. The MQTT broker acts as the central communication hub, allowing devices (clients) to connect, subscribe to specific topics, and publish messages to the broker. The broker then distributes the messages to all clients who subscribe to the relevant topic. By acting as a mediator, the broker decouples the devices, allowing them to work independently, and stores messages if a client is temporarily unavailable. MQTT brokers are crucial in enabling communication between IoT devices by providing a simple, reliable, and scalable message exchange solution (Bellavista & Zanni, 2016).

Figure 32 shows the topics each device subscribes to and publishes during the program operation, with only two tracking systems. However, it gives us an understanding of how the system would function with more tracking devices. By analysing the figure, we can see all the topics that are subscribed and published, but it is unclear what is included in the actual publishing packets. Table 5 presents the information transmitted through the topics.

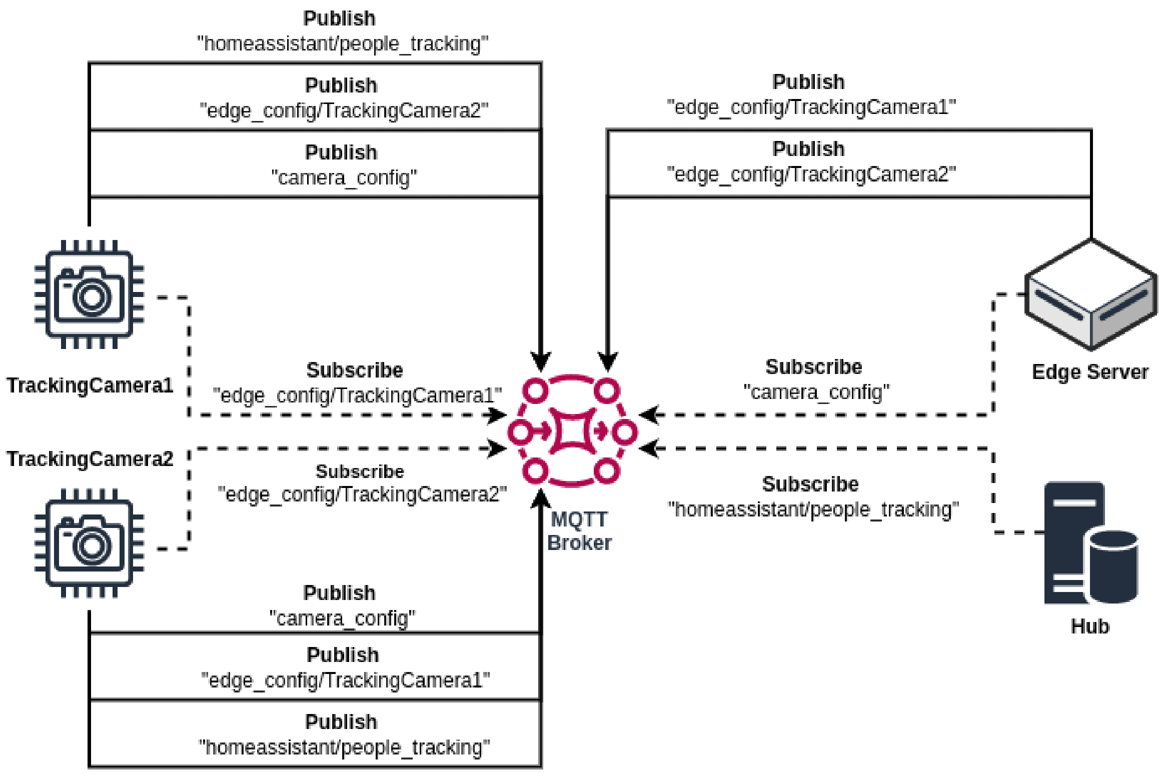

Publish "homeassistant/people_tracking"
Publish "edge_config/TrackingCamera2"
Publish "camera_config"
Publish "edge_config/TrackingCamera1"
Publish "edge_config/TrackingCamera2"
TrackingCamera1
Subscribe "edge_config/TrackingCamera1"
TrackingCamera2
Subscribe "edge_config/TrackingCamera2"
MQTT Broker
Subscribe "camera_config"
Edge Server
Subscribe "homeassistant/people_tracking"
Hub
Publish "camera_config"
Publish "edge_config/TrackingCamera1"
Publish "homeassistant/people_tracking"

Figure 32: MQTT Broker network connections in the system.

Table 5 MQTT Broker networks connections

Topic	**Devices Connected**	**Message content**
homeassistant/people_tracking	Tracking System $\rightarrow$ Hub	Final tracking information
edge_config/{TrackingSystem}	Tracking System $\leftrightarrow$ Tracking System	Re-identification request and re-identification acknowledgement
edge_config/{TrackingSystem}	Edge Server $\rightarrow$ Tracking System	Request to resend the configuration settings and send the new configurations
camera_config	Tracking System $\rightarrow$ Edge Server	Send configuration settings

6 Conclusion and future works

In conclusion, this dissertation has presented a novel framework for tracking and re-identifying people on a university campus as part of the Lab4U&Spaces project. By carefully selecting the appropriate technologies and architecture, It was able to demonstrate the viability of the proposed framework through experiments in a multi-camera environment. While the results are promising, there is room for improvement, including further testing with more cameras and real-world evaluations of the framework's behaviour in a dynamic campus environment. However, this framework can potentially provide valuable insights and drive new and more intelligent applications for campus management and life.

Going forward, it will be essential to continue exploring the potential of this framework by leveraging the rich data it can provide. This may include conducting additional experiments to evaluate its performance under different conditions and further exploring ways to optimise its efficiency and scalability. Additionally, it will be critical to engage with stakeholders, such as campus administrators, faculty, and students, to gather their feedback and insights into how this framework can best support the needs and goals of the campus community. Ultimately, this work represents a significant step forward in developing smart campus technologies and underscores the importance of continued innovation and collaboration in this field.

References

Abuarqoub, A., Abusaimeh, H., Hammoudeh, M., Uliyan, M., Abu-Hashem, M., Murad, S., ... Alfayez, F. (2017, 07). A survey on internet of things enabled smart campus applications. In (p. 1-7). doi: 10.1145/3102304.3109810

Aharon, N., Orfaig, R., & Bobrovsky, B.-Z. (2022). *Bot-sort: Robust associations multi-pedestrian tracking.* arXiv. Retrieved from `https://arxiv.org/abs/2206.14651` doi: 10.48550/ARXIV.2206.14651

Akkas, S., Maini, S. S., & Qiu, J. (2019). A fast video image detection using tensorflow mobile networks for racing cars. In *2019 ieee international conference on big data (big data)* (p. 5667-5672). doi: 10.1109/BigData47090.2019.9005689

Alghamdi, A., & Shetty, S. (2016). Survey toward a smart campus using the internet of things. In *2016 ieee 4th international conference on future internet of things and cloud (ficloud)* (p. 235-239). doi: 10.1109/FiCloud.2016.41

Alghamdi, A., Thanoon, M., & Alsulami, A. (2019, 01). Toward a smart campus using iot: Framework for safety and security system on a university campus. *Advances in Science, Technology and Engineering Systems Journal*, *4*. doi: 10.25046/aj040512

Almasawa, M. O., Elrefaei, L. A., & Moria, K. (2019). A survey on deep learning-based person re-identification systems. *IEEE Access*, *7*, 175228-175247. doi: 10.1109/ACCESS.2019.2957336

Arasteh, H., Hosseinnezhad, V., Loia, V., Tommasetti, A., Troisi, O., Shafie-khah, M., & Siano, P. (2016). Iot-based smart cities: A survey. In *2016 ieee 16th international conference on environment and electrical engineering (eeeic)* (p. 1-6). doi: 10.1109/EEEIC.2016.7555867

Azhar, M. I. H., Zaman, F. H. K., Tahir, N. M., & Hashim, H. (2020). People tracking system using deepsort. In *2020 10th ieee international conference on control system, computing and engineering (iccsce)* (p. 137-141). doi: 10.1109/ICCSCE50387.2020.9204956

Banerjee, S., Ashwin, T. S., & Guddeti, R. M. R. (2019). Automated parking system in smart campus using computer vision technique. In *Tencon 2019 - 2019 ieee region 10 conference (tencon)* (p. 931-935). doi: 10.1109/TENCON.2019.8929357

Bellavista, P., & Zanni, A. (2016, 09). Towards better scalability for iot-cloud interactions via combined exploitation of mqtt and coap. In (p. 1-6). doi: 10.1109/RTSI.2016.7740614

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. In *2016 ieee international conference on image processing (icip)* (p. 3464-3468). doi: 10.1109/ICIP.2016.7533003

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *Yolov4: Optimal speed and accuracy of object detection.*

Brocke, J. v., Hevner, A., & Maedche, A. (2020, 09). Introduction to design science research. In (p. 1-13). doi: 10.1007/978-3-030-46781-4_1

Challa, S., Wazid, M., Das, A. K., Kumar, N., Goutham Reddy, A., Yoon, E.-J., & Yoo, K.-Y. (2017). Secure signature-based authenticated key establishment scheme for future iot applications. *IEEE Access*, *5*, 3028-3043. doi:

10.1109/ACCESS.2017.2676119

Clarke, R. Y. (2013). Smart cities and the internet of everything: The foundation for delivering next-generation citizen services. *IDC Government Insights*, 1-18. Retrieved from http://www.cisco.com/web/strategy/docs/scc/ioe_citizen_svcs_white_paper_idc_2013.pdf

Computer vision. (2023, Feb). Wikimedia Foundation. Retrieved from https://en.wikipedia.org/wiki/Computer_vision

Dendorfer, P., Ošep, A., Milan, A., Schindler, K., Cremers, D., Reid, I., ... Leal-Taixé, L. (2020). *Motchallenge: A benchmark for single-camera multiple target tracking.* arXiv. Retrieved from https://arxiv.org/abs/2010.07548 doi: 10.48550/ARXIV.2010.07548

Du, Y., Song, Y., Yang, B., & Zhao, Y. (2022). *Strongsort: Make deepsort great again.* arXiv. Retrieved from https://arxiv.org/abs/2202.13514 doi: 10.48550/ARXIV.2202.13514

Fortes, S., Santoyo Ramón, J., Palacios Campos, D., Baena, E., Mora-García, R., Medina, M., ... Barco, R. (2019, 03). The campus as a smart city: University of málaga environmental, learning, and research approaches. *Sensors*, *19*, 1349. doi: 10.3390/s19061349

Fu, D., Chen, D., Bao, J., Yang, H., Yuan, L., Zhang, L., ... Chen, D. (2020). *Unsupervised pre-training for person re-identification.* arXiv. Retrieved from https://arxiv.org/abs/2012.03753 doi: 10.48550/ARXIV.2012.03753

Gai, Y., He, W., & Zhou, Z. (2021). Pedestrian target tracking based on deepsort with yolov5. In *2021 2nd international conference on computer engineering and intelligent control (icceic)* (p. 1-5). doi: 10.1109/ICCEIC54227.2021.00008

Gil, D., Ferrández, A., Mora, H., & Peral, J. (2016, 07). Internet of things: A review of surveys based on context aware intelligent services. *Sensors*, *16*, 1069. doi: 10.3390/s16071069

Hu, H., Hachiuma, R., Saito, H., Takatsume, Y., & Kajita, H. (2022). Multi-camera multi-person tracking and re-identification in an operating room. *Journal of Imaging*, *8*(8). Retrieved from https://www.mdpi.com/2313-433X/8/8/219 doi: 10.3390/jimaging8080219

Joshi, S., Saxena, S., Godbole, T., & Shreya. (2016a). Developing smart cities: An integrated framework. *Procedia Computer Science*, *93*, 902-909. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050916315022 (Proceedings of the 6th International Conference on Advances in Computing and Communications) doi: https://doi.org/10.1016/j.procs.2016.07.258

Joshi, S., Saxena, S., Godbole, T., & Shreya. (2016b). Developing smart cities: An integrated framework. In (Vol. 93). doi: 10.1016/j.procs.2016.07.258

Kramp, T., Kranenburg, R., & Lange, S. (2013, 09). Introduction to the internet of things. In (p. 1-10). doi: 10.1007/978-3-642-40403-0_1

Leng, Q., Ye, M., & Tian, Q. (2020). A survey of open-world person re-identification. *IEEE Transactions on Circuits* *and Systems for Video Technology*, *30*(4), 1092-1108. doi: 10.1109/TCSVT.2019.2898940

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., ... Dollár, P. (2014). *Microsoft coco: Common objects in context.* arXiv. Retrieved from `https://arxiv.org/abs/1405.0312` doi: 10.48550/ARXIV.1405.0312

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T.-K. (2021, apr). Multiple object tracking: A literature review. *Artificial Intelligence*, *293*, 103448. Retrieved from `https://doi.org/10.1016%2Fj.artint.2020.103448` doi: 10.1016/j.artint.2020.103448

Mohan, A., Kaseb, A. S., Gauen, K. W., Lu, Y.-H., Reibman, A. R., & Hacker, T. J. (2018). Determining the necessary frame rate of video data for object tracking under accuracy constraints. In *2018 ieee conference on multimedia information processing and retrieval (mipr)* (p. 368-371). doi: 10.1109/MIPR.2018.00081

Muhamad, W., Kurniawan, N. B., Suhardi, & Yazid, S. (2017). Smart campus features, technologies, and applications: A systematic literature review. In *2017 international conference on information technology systems and innovation (icitsi)* (p. 384-391). doi: 10.1109/ICITSI.2017.8267975

Redmon, J., & Farhadi, A. (2018). *Yolov3: An incremental improvement.* arXiv. Retrieved from `https://arxiv.org/abs/1804.02767` doi: 10.48550/ARXIV.1804.02767

Ristani, E., Solera, F., Zou, R. S., Cucchiara, R., & Tomasi, C. (2016). *Performance measures and a data set for multi-target, multi-camera tracking.* arXiv. Retrieved from `https://arxiv.org/abs/1609.01775` doi: 10.48550/ARXIV.1609.01775

Ristani, E., & Tomasi, C. (2018). Features for multi-target multi-camera tracking and re-identification. *CoRR*, *abs/1803.10859*. Retrieved from `http://arxiv.org/abs/1803.10859`

Soviany, P., & Ionescu, R. T. (2018). *Optimizing the trade-off between single-stage and two-stage object detectors using image difficulty prediction.* arXiv. Retrieved from `https://arxiv.org/abs/1803.08707` doi: 10.48550/ARXIV.1803.08707

Syed, A., Sierra-Sosa, D., Kumar, A., & Elmaghraby, A. (2021, 03). Iot in smart cities: A survey of technologies, practices and challenges. *Smart Cities*, *4*, 429-475. doi: 10.3390/smartcities4020024

Toutouh, J., & Alba, E. (2022). A low cost iot cyber-physical system for vehicle and pedestrian tracking in a smart campus. *Sensors*, *22*(17). Retrieved from `https://www.mdpi.com/1424-8220/22/17/6585` doi: 10.3390/s22176585

Vaigandla, K., Radha, K., & Allanki, S. R. (2021, 09). A study on iot technologies, standards and protocols.. doi: 10.17697/ibmrd/2021/v10i2/166798

Vaigandla, K. K., Karne, R. K., & Rao, A. S. (2021). A study on iot technologies, standards and protocols. *IBMRD's Journal of Management & Research*, *10*(2). Retrieved from `https://www.indianjournalofmanagement.com/index.php/ibmrd/article/view/166798`

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2021). Scaled-yolov4: Scaling cross stage partial network. In *2021* *ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 13024-13033). doi: 10.1109/CVPR46437.2021.01283

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). *Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.* arXiv. Retrieved from `https://arxiv.org/abs/2207.02696` doi: 10.48550/ARXIV.2207.02696

Wang, X. (2013). Intelligent multi-camera video surveillance: A review. *Pattern Recognition Letters*, *34*(1), 3-19. Retrieved from `https://www.sciencedirect.com/science/article/pii/S016786551200219X` (Extracting Semantics from Multi-Spectrum Video) doi: https://doi.org/10.1016/j.patrec.2012.07.005

What is computer vision? (n.d.). Retrieved from `https://www.ibm.com/topics/computer-vision`

What is the internet of things (iot)? (n.d.). Retrieved from `https://www.oracle.com/pt/internet-of-things/what-is-iot/`

Wieczorek, M., Rychalska, B., & Dabrowski, J. (2021). *On the unreasonable effectiveness of centroids in image retrieval.* arXiv. Retrieved from `https://arxiv.org/abs/2104.13643` doi: 10.48550/ARXIV.2104.13643

Wojke, N., & Bewley, A. (2018). Deep cosine metric learning for person re-identification. In *2018 ieee winter conference on applications of computer vision (wacv)* (pp. 748–756). doi: 10.1109/WACV.2018.00087

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 ieee international conference on image processing (icip)* (pp. 3645–3649). doi: 10.1109/ICIP.2017.8296962

Wu, H., Han, H., Wang, X., & Sun, S. (2020). Research on artificial intelligence enhancing internet of things security: A survey. *IEEE Access*, *8*, 153826-153848. doi: 10.1109/ACCESS.2020.3018170

Wu, M., Qian, Y., Wang, C., & Yang, M. (2021). A multi-camera vehicle tracking system based on city-scale vehicle re-id and spatial-temporal information. In *2021 ieee/cvf conference on computer vision and pattern recognition workshops (cvprw)* (p. 4072-4081). doi: 10.1109/CVPRW53098.2021.00460

Ye, J., Yang, X., Kang, S., He, Y., Zhang, W., Huang, L., ... Tan, X. (2021). A robust mtmc tracking system for ai-city challenge 2021. In *2021 ieee/cvf conference on computer vision and pattern recognition workshops (cvprw)* (p. 4039-4048). doi: 10.1109/CVPRW53098.2021.00456

Ye, M., Shen, J., Lin, G., Xiang, T., Shao, L., & Hoi, S. C. H. (2020). *Deep learning for person re-identification: A survey and outlook.* arXiv. Retrieved from `https://arxiv.org/abs/2001.04193` doi: 10.48550/ARXIV.2001.04193

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., ... Wang, X. (2021). *Bytetrack: Multi-object tracking by associating every detection box.* arXiv. Retrieved from `https://arxiv.org/abs/2110.06864` doi: 10.48550/ARXIV.2110.06864

Zheng, L., Tang, M., Chen, Y., Zhu, G., Wang, J., & Lu, H. (2021). Improving multiple object tracking with single object tracking. In *2021 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 2453-2462). doi: 10.1109/CVPR46437.2021.00248

Zhou, K., & Xiang, T. (2019). Torchreid: A library for deep learning person re-identification in pytorch. *arXiv preprint arXiv:1910.10093*.

Zhou, K., Yang, Y., Cavallaro, A., & Xiang, T. (2019). Omni-scale feature learning for person re-identification. In *Iccv.*

Zhou, K., Yang, Y., Cavallaro, A., & Xiang, T. (2021). Learning generalisable omni-scale representations for person re-identification. *TPAMI*.

Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). *Object detection in 20 years: A survey.* arXiv. Retrieved from `https://arxiv.org/abs/1905.05055` doi: 10.48550/ARXIV.1905.05055