Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea Magistrale in Data Science

Video Analytics for Understanding Pedestrian Mobility Patterns in Public Spaces: The Case of Milan

Relatore: *Prof. Michele Ciavotta*
Correlatrice: *Dott.ssa Giulia Ceccarelli*

Tesi di Laurea Magistrale di:
Lorenzo Lorgna
Matricola 829776

Anno Accademico 2021–2022

To my family.

Abstract

Nowadays, thanks to the recent developments of ICT tools for collecting traffic data in the urban environment, there is ever-growing availability of videos regarding the nerve centres of cities. This enables detailed analyses of public spaces and their users, for a better understanding of hidden mobility patterns. Having a clear view of pedestrian dynamics can be very helpful for various applications in modern days such as urban planning.

The main objective of this research project is to characterise public spaces through a mobility study on pedestrian patterns analysed by means of video analytics. The analysis focuses on the different types of pedestrian dynamics and on pedestrian route choices to obtain pedestrian utilisation profiles through observable behavioural parameters.

This work revolves around the case study of Piazza Duomo (Milan, Italy). A tracking-by-detection approach was chosen for the analyses; YOLOv7 was used to detect pedestrians after being trained on a custom dataset and, in order to handle pedestrian tracking, SORT algorithm was considered.

The distribution of detections was analysed using QGIS software, whereas, point pattern analysis and trajectory data mining were performed using specific Python libraries. The trajectory data mining part was aimed at clustering the trajectories in order to identify pedestrian utilisation profiles (namely, commuters and tourists). Secondly, group analysis was carried out to identify groups of pedestrians that might be representative of specific mobility patterns in the public space considered.

The results of the study show that conducting analysis in crowded scenes can be challenging, both with a human eye and with video analytics techniques. Nonetheless, computer vision techniques can provide great advantages, especially in speeding up the overall process. Within the Urban Informatics discipline, the results of this research could support the definition of evidence-based/ parametric approach for regeneration projects of urban public spaces. Moreover, the proposed methodology could be of notable interest for the calibration/ validation of computer-based pedestrian modelling and simulation systems, and for further development of computer vision techniques.

Contents

List of Figures

List of Tables

Acknowledgement

This research has been carried out during a curricular internship at Fondazione Transform Transport ETS (Milan, Italy), the non-profit research foundation launched by Systematica Srl on March 2022 and focused on innovation in mobility and transport planning. The analysed data were treated according to the General Data Protection Regulation (EU, 2016/679). This research received no specific grant from any funding agency in the public, commercial and not-for-profit sectors.

The results of this research have been accepted for oral presentation at the International Conference on Pedestrian and Evacuation Dynamics (PED 2023) (Lorgna et al., 2023a) and submitted for oral presentation to the European Transport Conference (ETC 2023) (Lorgna et al., 2023b).

I would like to express my gratitude to Prof. Michele Ciavotta, Giulia Ceccarelli and Dr. Andrea Gorrini for their supervision and collaboration.

I would also like to extend my thank to my family, Chiara, and all of my friends who have been a constant presence and source of support throughout my academic journey.

Chapter 1

Introduction

The growing availability of data and an increasing computational capacity is enabling significant growth in different fields, including Urban Informatics. This is an innovative field in which computer science and data science techniques converge with urban planning concepts on geo-referred data. Urban planners in this scenario employ new technologies and new methods to solve traditional urban planning problems exploiting the new potential of computation. Big Data is indeed changing the way how to collect, analyse, and interpret mobility patterns providing new approaches to quantifying public spaces and related human dynamics. Unprecedented data volumes therefore require the use of new techniques and the exploitation of computational power.

Urban Informatics captures the intricate nature of urban space and its dynamics. This is an evidence-based approach that leads to innovative assessment tools and metrics useful for investigating mobility patterns in urban spaces. The success of deep learning and computer vision has created opportunities and laid the foundation for understanding better cities through images. In particular video analytics techniques through the evolution of machine learning and deep learning have introduced the automation of tasks that were once the exclusive purview of humans who manually identify and study different relevant behaviours in the scenes.

> "*This revolution in tracking human and other motion in digital form enables the collection of multiple attributes at the finest of scales of urban observation.*"(Batty, 2010)

Understanding the dynamics of human behaviour and more particularly human mobility has been one of the emerging interests among researchers, considering also changes in Government policy more concerned about the impact of growing car populations in urban areas and the environment and sustainability as mentioned in (Derrick, 1999). People have different lifestyles and cities are getting more complex and bigger every day: these two aspects make the analysis of human mobility challenging. In particular, walking among the various modes of transport while on the one hand it can be considered as one of the

most natural, on the other hand, from the analyst's point of view, it is actually characterised by complex aspects. One of these is the fact that considering also other modes of transport in the case of walking, there is no vehicle associated with it and the context in which this mode takes place is heterogeneous and consists of different elements such as sidewalks, crossings, etc. as described in (Bierlaire and Robin, 2009).

To provide a more detailed definition of Urban Informatics, it can be said that it originates from the encounter and interaction of three different domains in urban environments as described in (Foth et al., 2011): People, Places, and Technologies (see Figure 1.1). Places are a starting point of urban analysis. In

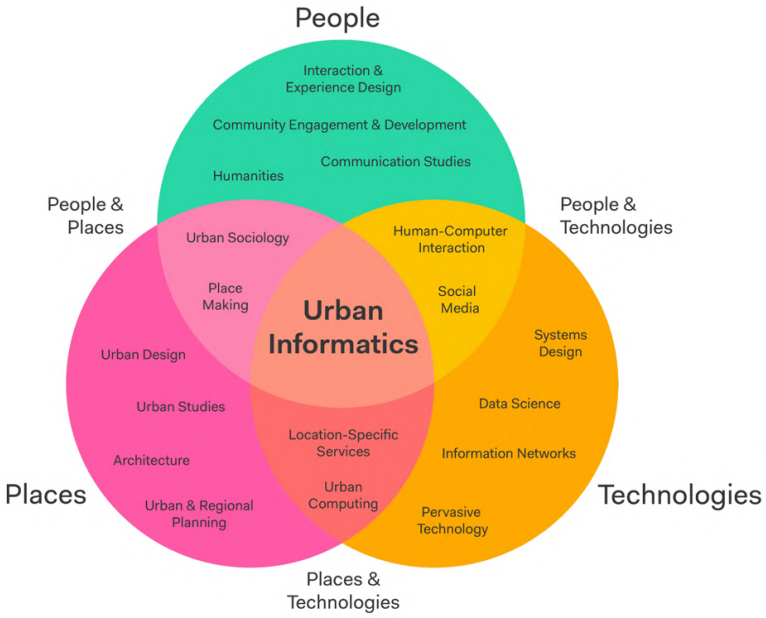


Figure 1.1: Urban Informatics relational chart. Image is adapted from (Foth et al., 2011).

recent years, urban densification has affected the entire world, resulting in an increasingly large number of people living in cities. By 2050, 66% of the world's population will live in cities (United Nations, 2014). There is consequently a need now more than ever to obtain a detailed understanding of the spaces that characterise cities. People are the core element on which urban analyses are based. The relationship between them and urban spaces needs to be explored. These kinds of analyses are supported by the use of innovative technologies that have an important role, from collecting data to analysing it.

The aim of the study is to transform raw data from observations into meaningful information, relevant to urban planning considerations. The dataset that will be used is related to the videos of the webcam in Piazza Duomo in Milan (Italy). Object detection and object tracking models will be used to identify pedestrians and track their movements within the square. This work investigates whether these innovative techniques can really provide advantages in

the field of urban planning and mobility. More specifically, whether through the results of computer vision and deep learning techniques it is possible to perform a mobility study of a public space, analysing the distribution of the people identified and their trajectories using specific urban planning metrics. An attempt will be made to identify also different categories of pedestrians (commuters and tourists) and investigate the presence or absence of groups of people.

The thesis is organised as follows. Chapter 2 provides a theoretical background on urban mobility concepts and computer vision techniques, focusing on object detection and object tracking. Chapter 3 outlines the methodology carried out in this research and gives information on the datasets used for training the object detector and for the analysis phase. In addition, the main tools used in this research are presented. Chapter 4 illustrates the results of the experiments conducted together with a discussion to interpret the meaning. Last, Chapter 5 summarises the main results obtained and provides critical answers to the proposed research questions. Then, it lists the main ideas on future work related to this field of study.

Chapter 2

Theoretical Background and Related Works

The Chapter presents a comprehensive overview of the underlying urban mobility concepts and computer vision techniques. A review of the main approaches to perform urban mobility studies and the major contributions concerning the use of innovative techniques in this field are presented. Finally, the Chapter addresses the limitations of the current study.

2.1 Urban Mobility Studies and Public Spaces

Cities can be defined as complex/ socio-technical systems (Hillier, 2012) characterised by the combination of physical (e.g., buildings, streets, infrastructure, etc.) and human factors (e.g., movement, interaction, activity, etc.). In this context, urban mobility study refers to the investigation of how people and goods move through a city. In particular, the analysis of the behaviour of pedestrian crowds is a topic of growing interest supporting an improved understanding of human behaviour for the purpose of planning and designing public spaces.

Since the last 20 years, public spaces became a relevant aspect in urban mobility and transport planning, thanks to their central role in the creation of inclusive communities. As a result, the relationship between public space and urban mobility becomes a key aspect in the effort of designing sustainable and liveable cities, with a focus on walkability (Wefering et al., 2013).

> "*Public spaces are not only places where activities take place; they are also places for mobility, for people to come to, leave from and pass through.*"(Ravazzoli et al., 2017, p.39)

Among the most relevant studies concerning the empirical understanding of how people relate to public spaces are those of William H. Whyte (1980). In his work, he illustrates how people use a public space through photographs, footage, and conversations. Other noteworthy publications on social life in

public spaces are those by Jan Gehl (1971; 2010). These researches focus on proposing a reinterpretation of cities from the perspective of people with an increased emphasis on the performance, vitality, and use of public spaces. The intent is to consider cities more concerned with human dynamics offering people-friendly environments that allow opportunities for social interaction.

Data represents a key starting point to support the study of pedestrian behaviour in urban settings, and there are different data collection methods that are frequently used for this purpose (Feng et al., 2021), such as: (*i*) unobtrusive observation of pedestrian behaviour in natural environments; (*ii*) experimental investigation of pedestrian movements in a controlled scenario; (*iii*) data collection about pedestrian dynamics by using a survey including a list of predetermined questions.

As mentioned in (Hou et al., 2020), the systematic observation of pedestrian dynamics is a very informative method to record what people do and how they behave in a particular space. In order to get such kind of records researchers can utilise manual or automated techniques for people counting and tracking (Messa et al., 2022), or exploit sensor records such as GPS (Lin and Hsu, 2014) and Wi-Fi (Gorrini et al., 2021; Farrokhtala et al., 2018). Traditional data collection methods as mentioned in (Zhanga et al., 2023) provide rich insights but require significant resources in terms of time and labour. Moreover, these techniques are useful in a limited scenario; on a large scale and over long periods of time, analyses become overly complex. These new monitoring and more innovative solutions can actively cover pedestrian behaviour with a larger spatial and time scale. In this framework, the current thesis work relies on field observations supported by camera sensors, considering their potential as a rich source of spatial-temporal information.

Characterising public space becomes the first step in analysing pedestrians and their relationship with it. This is meant to have a better understanding of pedestrian patterns and utilisation profiles in public spaces (e.g., local density, speed, trajectories, impact of urban attractors, layout of the environment, etc.). These are obtained by the analysis of pedestrian movements and route choices. In particular, according to (Gorrini et al., 2016), the current thesis work proposes the following conceptualisation of different pedestrian behaviours in public spaces:

- *Time-driven*: people who have time constraints and walk through a certain environment constantly adjusting the trajectory between origin and destination to preserve a high speed (namely, singles, and commuters accessing public transport services).
- *Space-driven*: tourists or shoppers who visit for the first time a certain environment or have an exploratory attitude. Sometimes they are organised in large groups led by a guide. They stop more often, either for taking pictures of interesting spots or for shopping.

- *Social-driven*: strollers and inhabitants who amble through a certain environment since they live or work nearby the area (namely, small groups or families). They can stop their walk for an improvised conversation with somebody they know or for looking at the shop windows.

2.1.1 Spatial Data Science

The work of this thesis will focus on Spatial Data Science, one key area within Urban Informatics, described in Chapter 1. Spatial Data Science is the discipline that results from the encounter of Geographic Information System (GIS) world with data science in order to interpret spatial data. Over the past few years, there have been different names for this new field but the main common aspects are geography, (spatial) statistics, and computer science. GIS is a system for creating, managing, visualising, and analysing data and its spatial component. Two popular applications for GIS are ArcGIS and QGIS. Data science refers to the main processing techniques based on math, statistics, computing, and domain knowledge. Data science leverages both machine learning and deep learning which are powerful techniques for automatically learning patterns and creating decision-making models. These concepts, GIS and data science, are therefore used in combination to solve geographically oriented problems focusing on statistical analysis of patterns. This use of data science is leading to a change in the way spatial analysis is done, with a greater focus on programming rather than traditional software. Thus, by exploiting the availability of urban big data and these new technologies, it is possible to capture insights and knowledge useful for decision-making processes as mentioned in (Boeing et al., 2022).

In this thesis, the focus will be on point pattern analysis in order to characterise public spaces and trajectory data mining to discover different pedestrian utilisation profiles. The aim of the study is therefore to transform raw movement observations into meaningful information.

Point Pattern Analysis Point pattern analysis refers to a collection of points that have geographical attributes and they show a particular distribution. Points represent the place where a particular event occurred and in this thesis, they indicate the location of identified pedestrians. It becomes relevant trying to capture the patterns that characterise point distribution in space. There are three main categories of point pattern analysis techniques: descriptive statistics, density-based methods, and distanced-based methods. The first approach consists in analysing these point patterns calculating descriptive statistics (centrography) and it can be useful to consider some graphs to have a better understanding. This is a simple approach but it can provide a quick overview of the case as a whole. More powerful methods, density-based methods, and distanced-based methods can be used to investigate patterns more deeply. Density-based methods focus on first-order properties of the dataset while distanced-based methods consider second-order properties such as interactions between points.

Trajectory Data Mining Trajectories represent the paths of pedestrians in a given geographical area and they are therefore characterised by spatial and temporal information. A trajectory can be seen as a sequence of chronologically ordered spatio-temporal points: $p_1 \rightarrow p_2 \rightarrow \cdots p_n$ where each point p consists of a geo-spatial coordinate set and a timestamp, (x, y, t). With recent technological advances, the amount of datasets with spatio-temporal data has increased and this is referred as trajectory data mining (Zheng, 2015). The survey mentioned presents the various aspects that constitute trajectory data mining: trajectory preprocessing, trajectory indexing and retrieval, and different data mining tasks, like trajectory pattern mining, trajectory uncertainty, outlier detection, and classification. Trajectory preprocessing is one of the first steps required to process trajectories and perform mining tasks. It involves operations such as noise filtering, segmentation, compression, map-matching, and stay-point detection. In order to eliminate noise there are some techniques such as considering a speed threshold between two consecutive points, setting a minimum threshold of points for each trajectory as described in (Idrissov, 2012). Trajectory indexing and retrieval, on the other hand, refers to the organisation of trajectory data in such a way that it can be used as efficiently as possible. As far as the last steps are concerned, which consist of a series of mining tasks, considering the trajectory pattern mining step, the aim is to analyse and identify patterns in trajectories and thus in pedestrian movements, e.g. through trajectory clustering which is one of the most intuitive and powerful techniques for gaining useful information and knowledge from trajectory data (Yu et al., 2019). Trajectory clustering as mentioned in (Bian et al., 2018) has different potentialities in different areas: object motion prediction, traffic monitoring, activity understanding, abnormal detection, 3-dimensional reconstruction, weather forecasting, and geography.

Considering clustering, based on the presence of labelled data, it is possible to group the existing techniques into three different types: supervised, semi-supervised, and unsupervised learning. Focusing on the latter type there are three main categories: partition-based, hierarchical-based, and density-based. The first consists of dividing the observations into k partitions called clusters. An iterative procedure is used to assign the observations to the best cluster. K-Means is one of the most widely used methods belonging to this type. The hierarchical approaches divide the dataset in such a way as to create a hierarchy of clusters, thus creating a dendogram. There are two possible strategies for creating these hierarchical structures: agglomerative (bottom-up) and divisive (top-down). An example of a hierarchical clustering algorithm is represented by BIRCH. The last type, density-based clustering, is based on the concept that a cluster can be considered as a dense region of points that is differentiated and separated by regions of low density. This approach is more robust to the presence of noise and outliers. Representative algorithms include DBSCAN and OPTICS. In general, clustering algorithms are based on the assumption that the observations examined and clustered are represented by multidimensional vectors where the dimensions are attributes. In the case of trajectory data, the attributes that can be calculated are different such as: the x- and

y-coordinates of the start and end point, the point in the middle of the path, the mean x- and y-coordinates, and the distances between the start and end points in the x- and y-dimensions as presented in (Andrienko et al., 2013. A review of different possible approaches to address clustering of trajectories is presented in (Yuan et al., 2017).

In addition to clustering, another interesting area of study is group detection since groups represent the basic element for the analysis of crowds and detecting groups in crowds is becoming an important issue in modern behaviour analysis as mentioned in (Solera and Calderara, 2013). The presence of groups of people can indeed have a significant influence on overall dynamics. In particular, as mentioned in (Cavallaro and Vizzari, 2022), groups exhibiting reduced walking speed and a tendency to form stable patterns tend to influence collective dynamics at the microscopic level. *"A group is defined as two or more people interacting to reach a common goal and perceiving a shared membership, based on both physical and social identity"* as stated in (Solera et al., 2015). Several studies have shown that pedestrian flows in crowd situations show the presence of groups. In particular two-member groups (dyads) represent the most frequent and basic interacting elements of crowds as mentioned in (Gorrini et al., 2016). There are three different approaches to deal with group detection as described in (Solera et al., 2016): group-based, joint individual-group, and individual-based (Solera et al., 2016).

2.1.2 Computer Vision

Computer vision is a field of artificial intelligence that is related to image processing and pattern recognition. These are methods that receive images or videos as input and attempt to analyse them in order to extract useful data. This rapid development of computer vision has been made possible by the results obtained in the field of deep learning, which is based on neural networks. Neural networks consist of several layers: the low-level layers, those closest to the input, analyse the raw data, the intermediate layers in charge of reprocessing the information by extracting features of increasing complexity, and finally the final layers responsible for providing the desired high-level information. The optimal representation of the output is learned from the samples used to train the model, without requiring the manual step of feature definition. At a high level, therefore, the aim of computer vision is to replicate the way human vision works through computational power in order to learn and extract high-level information from images.

Since the 1990s, the scientific community has started to show more interest in computer vision techniques such as object detection and object tracking applied to subjects such as pedestrians. In the following years, there have been many publications in this field. Nonetheless, as mentioned in (Brunetti et al., 2018), the volume of these publications focusing on the application of video analytics techniques on pedestrians over time has always been lower than the number of publications on car and public transport, considering transportation

and urban planning literature.

The underlying idea that combines the computer vision field with urban planning analysis is based on the attempt to exploit technology in combination with the availability of open data as mentioned in (Ceccarelli et al., 2023b)[1]. In particular, computer vision has made great progress in recent years, and through the use of algorithms such as object detection, object tracking, and object counting, it is possible to replicate the analyses described in Chapter 2.1.1 in a more or less efficient and definitely faster manner. There are many analyses that can be carried out using computer vision, such as crowd counting and density estimation, crowd motion detection, crowd tracking, and crowd behaviour understanding. A description of these is given in (Tripathi et al., 2019).

An example of computer vision techniques being used to study urban mobility is given by (Ceccarelli et al., 2023a)[2]. In this work, pedestrian and vehicular flows during the pre- and post-intervention phases of the area were analysed by applying computer vision techniques to videos obtained by CCTV cameras. The results obtained demonstrated that through the use of these techniques, it was possible to identify the successful and critical elements of intervention and to use these observations in the subsequent stages of urban space design. Another interesting contribution that deals with pedestrian trajectory analysis obtained from videos using computer vision techniques is (Niu et al., 2022) in which the public space vitality is analysed and quantified by means of the observation of five quantitative indicators: the number of people, the duration of staying, motion speed, trajectory diversity and trajectory complexity. (Ibrahim et al., 2020) represents an attempt to list different computer vision algorithms related to tasks in urban planning.

The main aim of this work is to show the potential of computer vision in understanding urban dynamics and therefore there will be a focus on the details of different computer vision techniques, such as object detection, and object tracking.

2.2 Object Detection

Object detection consists in detecting instances of objects of a certain class within an image, like pedestrians, cars, buses, faces, etc, and drawing a tight bounding box around them. In this thesis, images will be the consecutive frames that compose the videos provided and pedestrians are the subjects. Object detection is a two-step process, it indeed involves both localisation and classification tasks. The localisation task aims to identify where the potential location of each target object is localised while the classification task allows to

[1]Partially available at: https://transformtransport.org/research/urban-mobility-metrics/looking-with-machine-eyes-understanding-patterns-in-urban-spaces/.

[2]Partially available at: https://transformtransport.org/research/urban-mobility-metrics/video-analytics-for-the-assessment-of-street-experiments-the-case-of-bologna/.

determine which predefined class each object belongs to. The object detection models need to be trained with a huge amount of annotated visual data which in turn helps to process the information in the new data. A comprehensive review of the various methods of object detection was done by (Zou et al., 2023).

Other works presenting the field of object detection in detail are the following reviews: (Aziz et al., 2020), (Zhao et al., 2019), (Liu et al., 2020), (Jiao et al., 2019), (Tomè et al., 2016).

2.2.1 Traditional Object Detection Approaches

In traditional machine learning-based approaches, before the growth of deep learning, object detection methods were based on the identification of hand-crafted features, such as the colour histogram, edge information, and texture. The aim is to identify groups of pixels that in some way could refer to some of the features of the object being considered. These features that were manually designed were then fed into a regression model that predicts the location of the object along with its label. Until 2010, this approach was widely used.

Traditional object detection approaches usually have three main stages:

- *Informative region selection*: The first step refers to the attempt to find the object's location called region of interest (ROI). Since objects can have different sizes or aspect ratios and can be present in different parts of the image, a multiscale sliding window image scanning approach is used. However, this approach is resource-intensive and it produces many irrelevant candidates. The region proposal algorithm can be Objectness, Selective Search, or category-independent object proposal.
- *Feature vector extraction*: This second phase aims to extract visual features using different techniques such as Haar-like, SIFT, and HOG. These feature vectors obtained should represent the semantic and robust nature of the candidate region.
- *Classification*: Support Vector Machine (SVM), Deformable Part-based Model (DPM), or Adaboost is used for the last classification step. These region classifiers were learned to assign categorical labels to the covered regions in order to distinguish a target object from all the other categories present in the above-mentioned regions.

The most representative methods of this era are Viola-Jones (VJ) detectors, Histogram of oriented gradients (HOG), and deformable part models (DPM).

The VJ detector was proposed in 2001 to solve a specific task, real-time face detection. It starts transforming the image to grayscale and after that uses sliding windows to search for haar-like features, in order to see if any window contains a human face (Viola and Jones, 2001). Haar-like features can be of three types: edge features, line features, and quadrilateral features.

During the Conference on Computer Vision and Pattern Recognition held in 2005 a different method proposed by N. Dalal and B. Triggs, HOG detectors, gained popularity (Dalal and Triggs, 2005). This approach gave better results but at the same time it was much slower. HOG can be seen as an improvement to scale-invariant feature transformation and shape contexts. This method attempts to extract contrast in different regions of the image. It is based on the concept that local object appearance and shape within an image can be represented by distributions of intensity gradients or edge directions. These gradient vectors are constituted from the magnitude and direction of change in the intensities of the pixel within a certain block of the image.

A third approach was deformable part models, DPM, proposed by P. Felzenszwalb in 2008 as an extension of the HOG detector (Felzenszwalb et al., 2010). This method follows the detection principle of *"divide and conquer"* indeed the training can be seen as learning the process to decompose an object into its main parts, and the inference phase can be regarded as ensembling detections of the different parts constituting the object.

Object detection reached a plateau after 2010 as the performance of these hand-crafted feature methods became saturated. Moreover, clear limitations of these methods were beginning to emerge as mentioned in (Wu et al., 2020b). In the first phase of region proposal generation using sliding windows techniques many proposals were generated and many of them turn out to be redundant. These techniques were designed manually and following heuristic approaches, therefore the accuracy in many cases could be lacking. In addition, these feature descriptors were hand-crafted based on low-level visual cues. This made it complicated to obtain a semantically accurate description of objects in more or less complex scenarios. Finally, considering these methods, the steps which constitute the pipelines of these approaches are often conceived and designed separately, which does not allow for a good overall solution of the system.

2.2.2 Deep Learning Object Detection Approaches

A revolution came in 2012 as mentioned in (Liu et al., 2020) with AlexNet. AlexNet is a Deep Convolutional Neural Network (CNN) architecture that won the ImageNet competition championship. Since that moment more and more studies have focused on object detection methods based on deep learning and in particular CNN. Before that time CNN saw heavy use only in the 1990s but then fell out of fashion by SVM which dominated until the 2000s. The reasons why deep learning had not yet taken over at that time were: lack of large amounts of annotated data leading to the problem of overfitting, limited computational resources, and less theoretical robustness than SVM. But it was clear that over time notable improvements could have been achieved with neural networks and soon deep learning methods become the state-of-the-art approaches to object detection.

Deep learning-based approaches employ CNNs to perform object detection, in

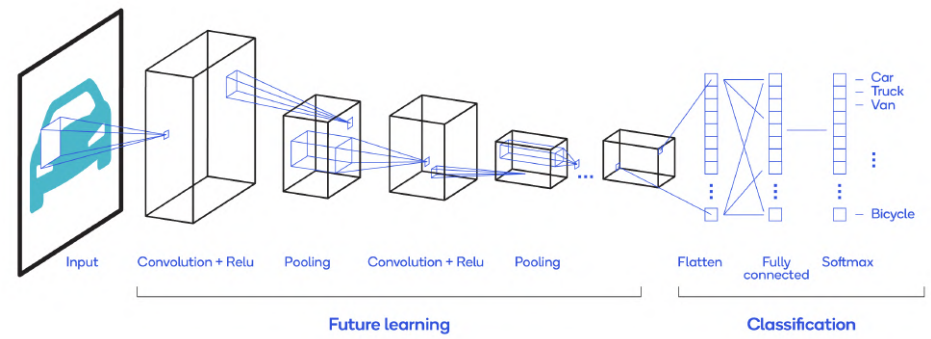


Figure 2.1: CNN architecture

which features don't need to be defined and extracted separately as it was for traditional approaches previously presented. The structure of a convolutional neural network (see Figure 2.1) is similar to the general structure of an artificial neural network. It consists of several layers: an input layer, a combination of several convolutional and pooling layers, one or more fully connected layers, and finally an output layer. The convolutional layer is the fundamental block of a CNN where the most computational operations are executed. The pooling layer, on the other hand, has the task of reducing progressively the spatial dimension of the feature representation in such a way to have fewer parameters and reduce the cost of computation. Finally, the fully-connected layer represents the final part of the network architecture and groups the information obtained in the previous layers. These final values are passed to the output layer which will have the task of performing the classification using a special probabilistic function. In contrast to hand-crafted descriptors used in traditional detectors, deep convolutional neural networks have several advantages as described in (Zhao et al., 2019). They generate hierarchical feature representations from raw pixels to high-level semantic information. A deeper architecture with a larger dataset can provide also an increased expressive capability. Due to these advantages, the use of CNNs increased dramatically and the improved accuracy of results was clear.

Currently, deep learning-based object detection methods can be divided into two families, two-stage (also known as region-based) and one-stage (also known as unified) detectors. Two-stage detectors first use a proposal generator to generate a sparse set of proposals and extract features from each proposal, followed by region classifiers that predict the category of the proposed region. Two-stage detectors have high localisation and object recognition accuracy as well as good results for real-time object detection. On the other hand, better performance in terms of time can be achieved by one-stage detectors which do not make use of region proposals. There is a further approach that differs from these two previous ones and is called anchor-free detectors.

Two-Stage Detectors

(Girshick et al., 2014) proposed regions with convolutional neural network features, R-CNN, for multiple-object detection and this represents the beginning of the use of deep learning techniques applied to the object detection task. In fact, a simple CNN can be used for image classification and objection detection but the main limitation is that only one object at time can be detected and localised with a bounding box. With R-CNN multiple objects can be detected and some aspects of traditional methods can be improved: sliding windows are replaced with a selective search algorithm while the traditional hand-made feature extraction part is replaced with a convolutional neural network, which leads to better performances. R-CNN using Selective Search method generate around 1,000 - 2,000 regions (candidate bounding boxes) from an image, which are rescaled to a fixed resolution and then fed one by one to a CNN. The latter extracts a 4096-dimensional feature, and finally sends the extracted features to an SVM in order to classify the region and to a linear regressor to tighten the bounding box of the object, if such an object exists.

However, the CNN feature extraction step was applied to each region proposal independently, which made the network very slow. A new method was developed as a fast and more efficient algorithm, named Fast R-CNN, (Girshick, 2015). This method was devised when the same author of the previous approach, R-CNN, decided to deal with some of the drawbacks of R-CNN in order to build a faster object detection algorithm. In particular, Fast R-CNN enables to simultaneously train a detector and a bounding box regressor under the same network configurations, so a single Deep ConvNet is used for feature extractions while in R-CNN 2,000 ConvNets are used for each region of the image. The feature extraction step in fact is now performed before the region's proposal. This change saves disk space and improves the speed of the process. In addition, Fast R-CNN uses softmax for object classification instead of SVM which is used in R-CNN. According to the results obtained softmax slightly outperforms SVM for object classification.

Both of the above algorithms, R-CNN and Fast R-CNN, use selective search to generate the region proposals but this approach is slow and time-consuming. In order to overcome this issue, Shaoqing Ren et al. have proposed Faster R-CNN (Rezatofighi et al., 2019), an object detection algorithm that eliminates the selective search algorithm which constituted a bottleneck and lets the network learn the region proposals. Selective search algorithm was replaced with Region Proposal Network (RPN), a separate fast neural network. This innovative feature allows for significant improvements, especially in terms of time. Faster R-CNN consists of 2 modules: an RPN network for generating region proposals and a Fast R-CNN for detecting objects in the proposed regions. The main benefit of this framework is that the RPN shares the same convolutional layers with the object detection network, which reduces the detection time. This improvement represents the first near real-time deep learning detector.

One-Stage Detectors

The above-mentioned methods, R-CNN, fast R-CNN, and Faster R-CNN belong to the class of two-stage detection algorithms: they use heuristic methods such as Selective search, or CNN network to generate region proposal and then perform classification and regression on them. Other methods of this class are Mask R-CNN (He et al., 2017), SPPNet (He et al., 2015), and Pyramid Networks (Lin et al., 2017a). The second class of detectors, one-stage detectors, include various methods such as YOLO, SSD and RetinaNet. These one-stage detectors use a CNN network to directly predict the categories and positions of different targets. These methods, unlike the former, have no intermediate region proposal process, which is why one-stage methods are also called region-free detectors. In this way, object detection task can be seen as a regression problem. One-stage detectors are usually much faster but achieve less accurate solutions.

YOLO, (Redmon et al., 2016), represents the first one-stage detector in deep learning era. YOLO applies a single convolutional network to the full image in order to predict multiple classes and bounding boxes in one evaluation. The network focuses only on those parts of the image that have a higher probability of containing objects and does not look at the entire image as was the case with the approaches seen previously. This approach abandoned the previous detection two-step paradigm, proposal detection and verification, focusing on improving the speed of the object detector. Good results were obtained with real-time detection of full images and webcams. The YOLO family of models has continued to evolve since its initial release in 2016, developed in a custom framework called Darknet which is written in low-level languages. In particular, YOLOv2 and YOLOv3 are both by Joseph Redmon, YOLO creator, while YOLO models after YOLOv3 are written by new authors. These latest versions should not be considered sequential releases but as a result to varying goals based on the authors' whom released them. At this time the most recent version of YOLO is YOLOv8.

SSD, (Liu et al., 2016), was the second one-stage detector in deep learning era. Compared with previous approaches, SSD eliminates the need for the region proposal network and this allows for increased speed of operations. There are also two important improvements that allow SSD to maintain fairly high accuracy results while maintaining a high execution speed. One is that SSD uses multi-scale features and the other is represented by the use of default boxes, prior boxes with different scales, and aspect ratios. A modded version of SSD is DSSD which further improves SSD by using a larger network.

RetinaNet, (Lin et al., 2017b) as in the case of SSD was developed to achieve results in terms of accuracy similar to one-stage detectors but at a significantly higher execution speed. The lower accuracy values of the one-stage methods compared to the two-stage methods were, according to the researchers, due to extreme foreground-background class imbalance problems. It means that the positive and negative samples are not balanced. Therefore, to solve this

problem, the concept of Focal Loss was introduced, a loss function to reduce class imbalance in the training step.

Anchor-Free Detectors

The previous methods, one and two-stage detectors, belong to the class of anchor-based. Anchors, also called anchor boxes, are a set of pre-defined bounding boxes with various scales and ratios placed regularly on the feature maps. The last research highlights a new approach that is gradually emerging, anchor-free detectors whose aim is to simplify the object detection process. These methods get rid of the manual design of anchors and can represent bounding boxes in any scale and ratio. In this way, these anchor-free methods make it possible to avoid setting some of the hyper-parameters considered earlier and to achieve similar performance to anchor-based methods. This shows their potential in terms of generalisation ability as mentioned in (Zhang et al., 2020). This recent approach uses two kinds of methods to detect objects: keypoint-based methods and dense prediction methods. The former use several self-learned keypoints to describe bounding box defining the spatial extent of objects while the latter uses object centres to define positives and predict the four distances from them to the object boundary. An example of anchor-free detectors is CenterNet, (Fan et al., 2021), which considers objects as single points, predicting the X and Y coordinates of an object's centre and its extent (height and width).

2.2.3 YOLO

YOLO represents the first single-stage object detector approach. Its first version, created by Joseph Redmon, Ali Farhadi, and Santosh Divvala, was released in 2016 and it was inspired by the GoogleNet architecture. As an approach belonging to the one-stage method family, the object detection and classification steps are unified, therefore there is a single neural network that processes the overall image by predicting bounding box, probabilities, and class labels present in the input image in a single step.

Its main characteristic is therefore its speed, which has distinguished it from other methods from the beginning. Compared to other object detection models, YOLO makes more localisation errors, but at the same time has better generalisation, recognising fewer false-positives in the background of the image. The public availability of its source code is a further important aspect: it means that the community can constantly improve the model.

Since its release, different versions and variants of YOLO have been proposed, each providing a significant increase in performance and efficiency. The versions from YOLOv1 to YOLOv3 were created by Joseph Redmon and Ali Farhadi. YOLOv7 (Wang et al., 2022) is currently the most recent version of YOLO designed by the same authors who originally developed the YOLO architecture.

YOLO framework

The YOLO framework, shown in Figure 2.2, starts by dividing the image in $S*S$ cells and for each cell B bounding boxes are predicted, each one with a confidence score.

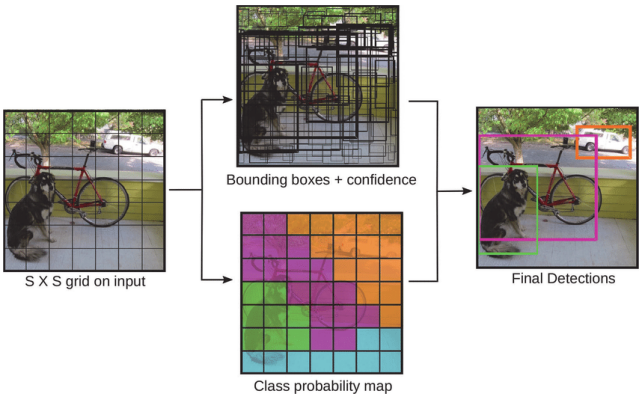


Figure 2.2: The overflow of YOLO. Image is taken from (Redmon et al., 2016).

The confidence score reflects how likely the box contains an object (objectness) and how accurate is the boundary box. It can be calculated using the formula: $\textit{Confidence score} = Pr(object) * IoU$. The first term, $Pr(object)$, Objectness Score, indicates the likelihood that a bounding box contains an object:

$$Pr(object) = \begin{cases} 1, & \text{if the centre of the object is considered inside the box.} \\ 0, & \text{otherwise.} \end{cases}$$

while IoU_{pred}^{truth} indicates how much the dimensions and localisation of the bounding box are correct with respect to the object. It reflects the Intersection of Union of the predicting bounding boxes b_{pred} and the ground truth b_{truth}.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

For each bounding box, the model predicts the following values: x, y, w, h, p. x and y coordinates represent the bounding box centre relative to the edges of the cell while its dimensions are w and h, respectively width and height. These values are normalised, all between 0 and 1. p indicates the box confidence score presented before.

Considering 5 components for each bounding box prediction, and the fact that each grid cell makes B of those predictions, there are in total S x S x B * 5 outputs related to bounding box predictions.

Every one of the grid cells also predicts, regardless of the number of the bounding boxes B generated, the conditional class probabilities, $Pr(Class_i|Object)$. This probability is conditioned on the grid cell containing one object and this makes sure that the loss function does not penalise a grid cell in the case where there are no objects present in it.

By multiplying the conditional class probabilities by the individual box confidence predictions seen before, confidence scores specific to each class for every bounding box can be obtained: $Pr(Class_i|Object) * Pr(Object) * IoU = Pr(Class_i) * IoU$. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

In case of multiple boxes for the same object, non-maximal suppression (NMS) technique becomes relevant. This procedure consists of some iterative steps that delete the bounding boxes that have a certain overlap value and keep only the best one. This procedure is useful because it is frequent that the model provides for two or more close cells bounding boxes which are very similar and that they are referring to the same object. For this reason, it is necessary to select only the better of the two.

YOLO architecture

YOLO architecture is FCNN (Fully Connected Neural Network) based. The YOLO framework has three main components: backbone, head, and neck. The backbone of a model is the element dedicated to taking the input image and extracting relevant features from it. This is a crucial step in any object detector, as it is the main structure responsible for extracting contextual information from the input image as well as for abstracting that information into patterns. The backbone component aims to extract these patterns similar to those learned during the training process. In this way, the image dimensionality is reduced. It can be trained from scratch but it is very expensive to do so. For this reason, the backbone network used for object detection is usually a deep convolutional network that was trained on an image classification task on a very large dataset, from which the last fully connected layer(s) is removed. The backbone feeds these features to the head through the neck. The objective of the neck is indeed aggregate as much information extracted by the backbone as possible before it is fed to the head. In particular, the neurons that constitute the neck component taking the previously processed image from the backbone are responsible for processing it in such a way as to extract the features required for class detection. The head is the part of the network that predicts the bounding boxes locations, the classes of the bounding boxes, and the objectness of the bounding boxes. It falls into two categories: two-stage detectors and one-stage detectors.

Focusing on YOLOv7 which is the latest official version of YOLO series, developed by YOLO architecture's original authors, this network further improves the detection speed and accuracy on the basis of the previous work. YOLOv7 was released in July 2022 by Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao and its source code has been released as open source under the GPL-3.0 license. Starting from YOLOv5, Pytorch was used instead of Darknet as a framework. This made it possible to speed up the training step and inference times than previous versions. The MS COCO dataset is used by the authors to train YOLOv7 without the use of any additional image datasets or pre-trained model weights. Compared to the other models presented, YOLOv7

model achieves a more accurate performance in the object detection step and at the same time it requires fewer resources for training than other models. On small datasets, the model is very fast even without pre-trained weights.

There are different versions of YOLOv7 including YOLOv7, YOLOv7-tiny, and YOLOv7-W6. There are additional versions, YOLOv7-X, YOLOv7-E6, and YOLOv7-D6, which allow for a deeper and wider model. The existence of a series of models makes it possible to meet the requirements of different applications, considering different levels of accuracy and speed

The architecture (see Figure 2.3) is derived from YOLOv4, Scaled YOLOv4, and YOLO-R but differently from those it uses PyTorch and not DarkNet as a framework.

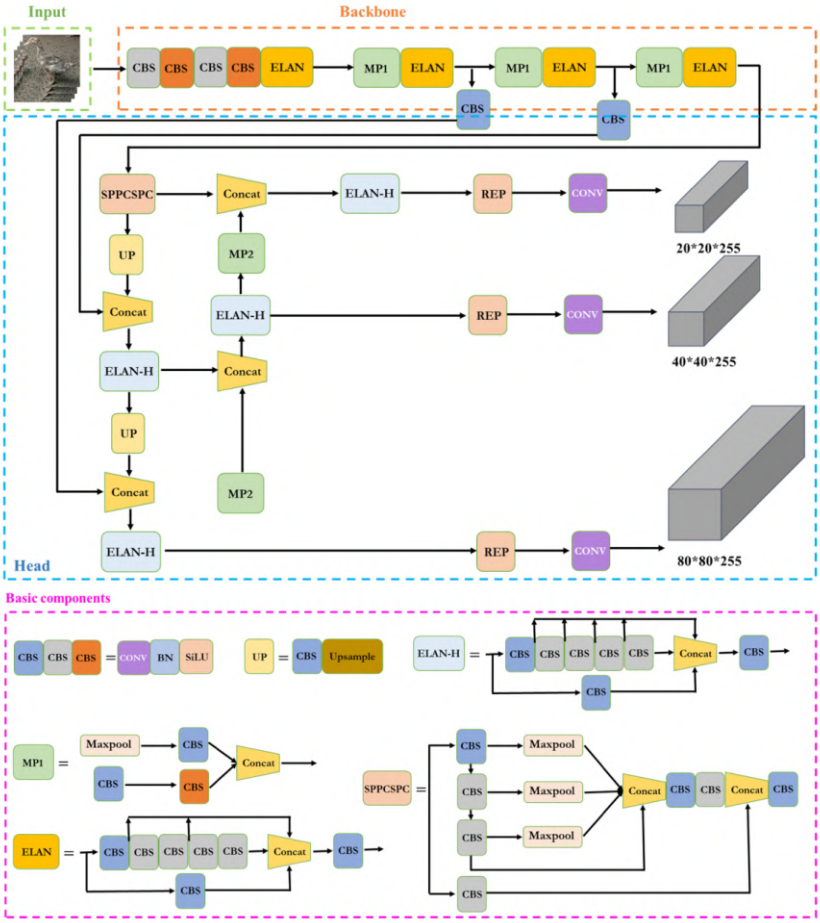


Figure 2.3: YOLOv7 network architecture diagram. Image is taken from (Jiang et al., 2022).

The most important aspects from an architectural perspective introduced by YOLOv7 are presented below. The innovative aspects introduced with YOLOv7 are architectural reforms (E-ELAN, Model Scaling for Concatenation-based Models) which enable better learning and trainable BoF (Planned re-parameterised convolution, Coarse for auxiliary and Fine for lead loss) which are methods

which increase the performance of a model without increasing the training cost.

Extended Efficient Layer Aggregation The Extended Efficient Layer Aggregation Network (E-ELAN) is the computational block in the YOLOv7 backbone. It takes inspiration from previous research on network efficiency. This component allows the model to continuously enhance the network's capacity for learning.

Model Scaling Techniques Depending on the objectives, there may be different needs. In some applications a very accurate model is required, in others a very fast model. Model scaling makes it possible to meet these requirements by providing great flexibility. There can be different types of scaling: width scaling (number of channels), depth scaling (number of stages), and resolution scaling (input image size).

Re-parameterisation Planning Re-parameterisation is a technique used after training to improve the inference results of the model. There are two types of re-parametrisation, model level and module level ensemble. In the first case, weights from multiple identical models with different training data are averaged. The latter is responsible for calculating a weighted average of model weights across various iteration numbers. These techniques make the model more robust to general patterns.

Auxiliary Head Coarse-to-Fine YOLOv7 inspired by Deep Supervision, a technique often used in training deep neural networks, makes use of an auxiliary head used to assist training in the middle layers in addition to the lead head, the head responsible for the final output.

Training, Transfer learning and Loss function

The training process of a neural network can be of two types, supervised and unsupervised learning. In the first case, the one considered in this thesis, the neural network is provided with annotated data, a training set consisting of inputs and related outputs. Using this information, the neural network tries to discover the relationship between the inputs and the outputs in such a way that it learns to generalise and can provide accurate results even in the presence of unannotated data. Therefore, training a model on a custom dataset allows for higher accuracy and precision for detection. Nevertheless, training a network from scratch requires a lot of computational and time resources. In order to overcome this problem, it is possible to train a neural network from pre-trained weights. This type of approach is called transfer learning and can be very advantageous before fine-tuning on a smaller custom dataset.

During the training step, the loss function assumes an important role since it is used to calculate the error between the prediction and the truth values of the inputs during the training phase. It is calculated as the sum of classification loss, localisation, and confidence loss. Classification loss is responsible for the classification task while the other two, localisation and confidence loss refer to

the detection task, i.e. the part related to the prediction of bounding boxes. During training, the aim is to minimise this function.

The YOLO loss function is presented below.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$
$$+\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_{ij} - \hat{C}_{ij} \right)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_{ij} - \hat{C}_{ij} \right)^2$$
$$+\sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Classification loss It indicates errors in the prediction's accuracy.

$$\sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

where $\mathbb{1}_{i}^{obj} = 1$ if an object appears in cell i, otherwise 0 and $\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

Localisation loss Responsible for predicting the correct position and size of bounding boxes. It indicates errors between the predicted boundary box and the ground truth.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

where $\mathbb{1}_{i}^{obj} = 1$ if the jth boundary box in cell i is responsible for detecting object, otherwise 0 and λ_{coord} increases the weight for the loss in the boundary box coordinates.

Confidence loss Responsible of giving a score to each of the bounding boxes so that all those that have a score below a certain threshold are then discarded. It is the objectness of the box or errors in the object's appearance in the bounding box.

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_{ij} - \hat{C}_{ij} \right)^2$$

where $\mathbb{1}_{i}^{obj} = 1$ if the jth boundary box in cell i is responsible for detecting the object, otherwise 0 and $\hat{C}_{ij}$ is the box confidence score of the box j in cell i. However, if an object is not detected:

$$\lambda_{\text{backg}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{backg}} \left(C_i - \hat{C}_i \right)^2$$

where $\mathbb{1}_{ij}^{\text{backg}}$ is the complement of $\mathbb{1}_{ij}^{obj}$, $\hat{C}_i$ is the box confidence score of the box j in cell i and λ_{backg} weights down the loss when detecting background.

YOLOv7 performances

As mentioned in (Wang et al., 2022) all the YOLOv7 models have better performances than the previous object detectors considering speed and accuracy in the range of 5 FPS to 160 FPS as seen in Figure 2.4.

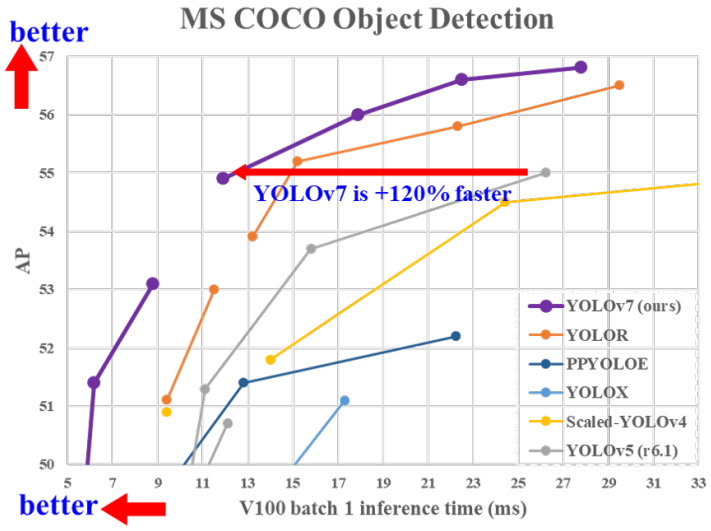


Figure 2.4: YOLOv7 perfomances. Image is taken from (Wang et al., 2022).

2.2.4 Object Detection Evaluation Metrics

Different metrics can be considered to evaluate the performance of an object detector. The most commonly used evaluation metric for object detection is the mean average precision (mAP), which is a high-level metric that is derived from a combination of Precision, Recall, and Intersection over Union (IoU). The inference time (ms) is also taken into account.

Precision Precision or specificity is defined as the ratio of True Positives to the total of True Positives and False Positives (incorrect identification as being

an object).

$$Recall = \frac{TP}{TP + FP}$$

Recall Recall or sensitivity is defined as the ratio of all True Positives (object is correctly identified) to the total of True Positives and False Negatives (missed instances of an object).

$$Recall = \frac{TP}{TP + FN}$$

IoU IoU score, seen before in 2.2.3, is used to measure how much of the predicted bounding box overlaps with the actual ground truth box. Therefore, it determines the correctness of a prediction and through it precision and recall are determined. It is mathematically defined as the ratio of the area of overlap between the two boxes to their combined areas. Usually, the threshold for IoU is kept as greater than 0.5. Although many researchers apply a much more stringent threshold like 0.6 or 0.7. If a bounding box has an IoU less than the specified threshold, that bounding box is not taken into consideration. The definition previously mentioned is:

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

mAP Average precision is defined as the area under the precision-recall curve. The values of this metric range from 0 to 1. mAP is the mean of the average precision for each class. mAP_0.5 is the mean average precision at an IoU threshold of 50% overlap between the predicted and ground truth bounding boxes while mAP_0.5:0.95 indicate the mean average precision over different IoU thresholds, ranging from 0.5 to 0.95.

$$mAP = \frac{\sum_{d=1}^{D} AP(d)}{D}$$

where D is the number of classes.

2.3 Object Tracking

Object tracking represents another important task in computer vision. Once the pedestrians are detected, these are tracked over consecutive frames. The object tracking task consists of generating trajectories for objects that are in motion in the available frames that constitute the video and assigning unique labels to them as seen in Figure 2.5.

The two fundamental concepts on which object tracking techniques are based are motion model and visual appearance model. The former consists in capturing and modelling the motion of the object. Examples of such techniques are optical flow, Kalman filtering, Kanade-Lucas-Tomashi (KLT) feature tracker, and mean shift tracking. On the other hand, visual appearance model is used to capture the appearance of the object that has to be tracked.

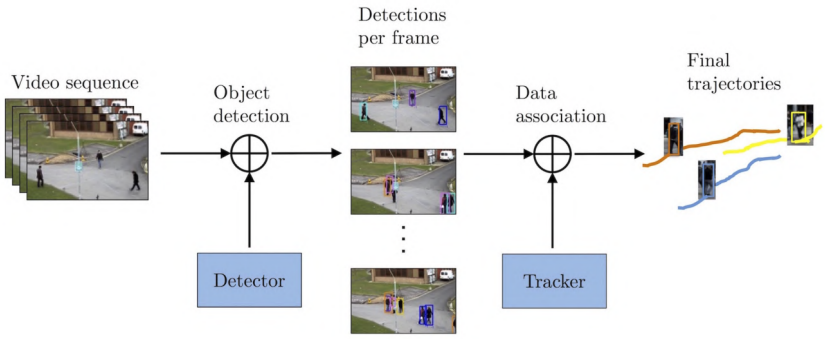


Figure 2.5: Tracking-by-detection paradigm. Image is taken from (Leal-Taixé, 2014).

Object tracking differs from object detection since the moment, during detection, one image at a time is considered and there is no information regarding the motion and past movement of the object, therefore it is not possible to uniquely track objects in a video. This is necessary to merge the knowledge of detecting objects in images with analysing temporal information and using it to predict movement trajectories. Unlike object detection, whose output is a set of bounding boxes in the image, object tracking also associates an ID to the bounding boxes in order to distinguish between detected objects of the same class. The association between a set of bounding boxes extracted from different frames and an ID forms the so-called tracklet.

Object tracking in this work is used to identify and track pedestrians in the several frames of the available videos with the purpose of extrapolating potential interesting patterns of space use. Another important pattern metric to better understand pedestrian dynamics is in fact represented by the trajectories. Traffic flow monitoring related to pedestrians is a relevant task that allows to describe the throughput of pedestrians in time and space between two locations. These pedestrians' trajectories can be obtained considering a tracking algorithm that receives in input the pedestrians' bounding boxes detected by an object detecotr model in an initial stage.

There are different types of object trackers. Depending on the number of objects that have to be tracked, single or multiple-object tracking (MOT) can be considered (see Figure 2.6).

In the case of a recorded stream, offline trackers are used while in online trackers future frames cannot be used to improve the results and the predictions are available immediately. In addition, online learning trackers learn about the object to track using the initialisation frame and a few subsequent frames. Instead, for offline learning trackers the training only happens offline, so these trackers don't learn anything during run time. An object tracker can be detection-free or detection-based. In the former case, a manual initialisation of a fixed number of objects in the first frame is required and it cannot deal with the case where new objects appear in the middle frames.

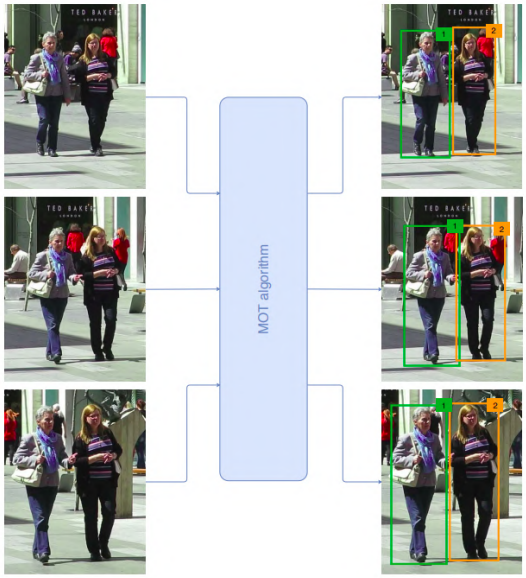


Figure 2.6: Multi-object tracking. Image is taken from (Ciaparrone et al., 2020).

In the latter case, consecutive video frames are given to a pre-trained object detector that gives detection hypothesis which in turn is used to form tracking trajectories.

Considering MOT algorithms the majority of these approaches perform either all or some of the following steps:

- *Detection stage*: an object detector is used to find all instances of objects of classes of interest in each video frame.
- *Feature extraction/ motion prediction stage*: the detections are analysed to extract features regarding appearance, motion, and interaction. A motion predictor is also used to predict the next position of each tracked target.
- *Affinity stage*: extracted features and motion predictions are used to compute a similarity/ distance score between pairs of detections and/ or tracklets.
- *Association stage*: the scores calculated in the previous step are used to associate detections and tracklets belonging to the same target using the same ID.

The current MOT framework can be divided into three types: MOT based on tracking-by-detection (DBT, also known as separate trackers), MOT based on joint detection and tracking (also known as joint trackers), and MOT based on attention mechanism. The first one is more widely used in industry and it is considered as standard approach employed in MOT algorithms as mentioned in the comprehensive survey (Ciaparrone et al., 2020). This approach first localises the target objects independently in each frame and later com-

bines information on appearance, motion, etc. with them with the aim to assign and connect these detected targets to existing trajectories. The object tracker is therefore consists of two components: an object detector and an embedding (re-ID) model. The inference time will be the sum of the computational times of the two components. The other approach, joint trackers, combines the detection and tracking phase in a single framework. Finally, MOT based on attention mechanisms represents a recent approach inspired by human perception. This type of model has shown promising results in challenging scenarios.

2.3.1 Traditional Object Tracking Approaches

Different methods have been implemented and can be grouped into two categories, traditional object-tracking approaches and deep-learning methods.

A lot of traditional tracking algorithms are built into the OpenCV tracking API. Most of these trackers are not very accurate compared to those that are based on machine learning. However, they can sometimes be useful to run in a resource-limited environment such as an embedded system.

One of these traditional tracking algorithms is Mean shift method (Fukunaga and Hostetler, 1975). It is an iterative algorithm based on the search for the mode of the distribution. This approach iteratively moves data points to a neighbouring region based on the average of the data points in that specific region. This idea is based on the concept that in correspondence with a movement of the object there is a variation of the distribution.

Continuously Adaptive Meanshift (CAMShift), (Bradski, 1998), is a colour-based object tracking method designed by Gary Bradski in 1998 which can be considered as an extension of the Mean shift model. This approach compared to the previous one achieves more accurate and more robust results. The CAMShift algorithm is analogous to Mean shift with the difference that it uses continuously adaptive probability distributions and it returns a resized and rotated target window. These two approaches, Mean shift and CAMShift, belong to the same category of object trackers as indicated in (Almohaimeed and Prince, 2019), kernel-based tracking.

Another approach that belongs in this category is Kanade-Lucas-Tomashi (KLT) feature tracking algorithm (Lucas and Kanade, 1981). Instead of tracking all the pixels that compose an image, this approach is based on the idea of tracking only a set of features. Specifically, in the initial stage, the model identifies trackable features and in subsequent frames attempts to track each of these based on its displacement.

Kalman filter (Kalman, 1960) is a different traditional method, belonging to the point-based tracking category. This approach starting from available detections and previous predictions tries to estimate the state of the system. Kalman filter is therefore a recursive estimator since the estimation of the

state of the system is based on information from the current and previous states.

2.3.2 Deep Learning Object Tracking Approaches

The overall accuracy of the above-mentioned traditional algorithms is low and they are mainly single-object tracking, which makes it difficult to meet the needs of complex scenes. In recent years, thanks to advances in deep learning, the results of tracking approaches in terms of accuracy have improved noticeably by using neural networks. In particular, among the various methods, tracking-by-detection approach has emerged as one of the most promising. More recently, joint frameworks based on detection and tracking and frameworks based on attention mechanism have also shown great potential with their performances.

Convolutional neural network-based offline training trackers represent one of the first tracking approaches that harness the power of convolutional neural networks to the task of visual object tracking. GOTURN, (Held et al., 2016), is one of the models that belongs to this category, based on convolutional neural network. The aim of the model is to track a given object from the given image crop, using both the current and the previous frame to accurately regress onto the object. The tracker is first trained for general object tracking and after that, it can be used to track most of the objects without any issues.

As seen before there are also online training trackers which use convolutional neural networks such as multi-domain network, MDNet, (Nam and Han, 2016), which was the winner of the Visual Object Tracking (VOT) 2015 challenge.

Another class of trackers with good performance based on Long Short Term Memory (LSTM) networks along with convolutional neural networks for the task of visual object tracking is Recurrent YOLO, ROLO, (Ning et al., 2017). It uses YOLO network for object detection while an LSTM network is used to find the trajectory of the target object.

A deep learning-based model known for its performance is DeepSORT (Wojke et al., 2017). It is an extension of SORT (Simple Online and Realtime Tracking) algorithm (Bewley et al., 2016). SORT was published in 2016 and it is based on two main components, the Kalman filter and Hungarian algorithms which allow to perform multiple object tracking. It can be considered as one of the first algorithms to handle object tracking in real-time. DeepSORT introduces deep learning by adding an appearance descriptor to reduce identity switches, hence making tracking more efficient. In DeepSort the bounding boxes are obtained using YOLO-v3. Then, based on Sort and ReID it is possible to link bounding boxes and tracks. If no link is identified between bounding boxes and tracks, a new ID is generated.

In addition, StrongSORT, (Du et al., 2023) equips DeepSORT with advanced components in various aspects.

Other novel approaches in the field of object tracking which belong to tracking-by-detection methods are FairMOT (Zhang et al., 2021) and ByteTrack (Zhang et al., 2022). This last one exploits YOLOX, a recent object detector in the YOLO series, to detect objects.

Considering joint-detection-and-tracking some notable methods are JDE (Wang et al., 2020), Tracktor++ (Bergmann et al., 2019), TrackR-CNN (Voigtlaender et al., 2019), CenterTrack (Zhou et al., 2020), PermaTrack (Tokmakov et al., 2021), TransTrack (Sun et al., 2020), and TrackFormer (Meinhardt et al., 2022).

2.3.3 SORT

SORT is a tracking algorithm designed to perform multiple object tracking belonging to tracking-by-detection approaches. To achieve real-time processing, SORT has a simple architecture. This approach uses the Kalman filter and the Hungarian algorithm (Kuhn, 1955) to track objects. The Kalman filter is an algorithm that can predict future positions based on the current position. The Hungarian algorithm can tell if an object in the current frame is the same as the one in the previous frame. It is used for association and ID attribution.

SORT SORT is a MOT algorithm based on detections and the entire process, as seen above, can be considered as the combination of these main steps: detection, estimation model, and object association which involves the creation and deletion of track identities.

Considering each new frame, SORT using the Kalman filter as a motion model try to propagate the already tracked objects. The Kalman filter performs this operation by extrapolating the motion of these objects and once performed updates its belief. Next, a two-stage or one-stage object detector detects all objects in the current frame.

These detections obtained will then be compared with the already tracked objects in order to be associated with the existing tracks. A cost matrix is thus created that evaluates the overlap (IOU) between predictions (obtained using the Kalman filter) and actual detections (obtained using the object detector). Using the Hungarian method object association is then performed: if the detection and the Kalman prediction overlap, the detection is assigned to the track while if a new detection has an IOU below a specific threshold it is not assigned to an existing track but classified as a new object and a new track will be created.

For simplicity and runtime, SORT has emerged as a baseline method renowned for its real-time speed and accuracy, achieving good performance at a high frame rate as mentioned in (Bewley et al., 2016). Certainly, this simplicity has its corresponding negative aspects, in particular, SORT due to a strong dependence on the detector's accuracy and occlusions suffers from tracking

loss and identity switching (IDS). This latter problem occurs since SORT, in order to guarantee high tracking efficiency, eliminates tracks that do not have a match in consecutive frames and this results in the creation of many new IDS.

DeepSORT Deep SORT is proposed as an approach to solve the issues presented above, tracking loss and IDS. To solve this, the model, considered as a successor to SORT, incorporates a Re-ID model in order to extract appearance features for each bounding box as described in (Wojke et al., 2017). These extracted features make it possible to match the predictions and detections for which occlusion events occur, thus reducing the IDS number.

The SORT and Deep-SORT methods share the same overall architecture providing a better association metric that combines both motion and appearance descriptors. Therefore, tracking is no longer based on the estimation of speed and position obtained by Kalman filter but in this case it is also based on the appearance of the object.

The first difference introduced by DeepSORT consists in the object association step: SORT uses exclusively the IOU-based distance while DeepSORT incorporates the IOU-based distance with a visual appearance distance. In particular, a weighted sum of the IOU-based distance and the visual appearance distance cosine between the visual appearance features of detections and already tracked objects is considered. DeepSORT also improves SORT's matching mechanism based on the IOU cost matrix. In fact, DeepSORT introduces the concept of Matching Cascade, whose aim is to give greater importance to objects that are most recently seen. Furthermore, DeepSORT unlike SORT which had no memory is able to store tracked objects and their appearance descriptors for up to 30 seconds. This becomes crucially relevant in presence of occlusions and different from SORT which completely loses the tracks, in this case there is the opportunity to restore it.

Due to these features, DeepSORT is one of the most well-known and widely used object-tracking approaches.

StrongSORT StrongSORT is a further improvement of DeepSORT. As described above, there is a combination of motion-based features and appearance-based features. However, these features are extracted by a more powerful feature extractor which is a neural network, more specifically a BoT with ResNeSt50 backbone and pre-trained on the DukeMTMCreID dataset. The use of this neural network makes it possible to obtain more discriminative features.

As mentioned by the authors of StrongSORT in (Du et al., 2023) this model despite great potential reveals some limitations. In particular, one of the most critical aspects concerns the execution time. In fact, the model turns out to be rather slow, especially when compared with joint trackers and appearance-free separate trackers.

2.3.4 Object Tracking Evaluation Metrics

As with object detection, a set of metrics can be identified for object tracking to assess the quality of the results obtained. Some of the most commonly used metrics include MOTA, IDF1, MOTP, and HOTA.

MOTA Metrics such as MOTA and IDF1 overemphasize detection and association respectively. MOTA measures tracking accuracy taking into account the ability to identify objects across frames, the ratio of false positives, and the mismatches. It is defined as:

$$MOTA = 1 - \frac{\sum(FN + FP + ID_S)}{\sum \text{Total objects in frame}}$$

Where FN is the number of false negatives, FP is the number of false positives, IDS is the number of identity switches at time t. MOTA can be negative too.

MOTP Another metric that can be used is Multiple Object Tracking Precision (MOTP) which measures the localisation error of the tracker for the matched targets and objects.

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}$$

MOTA and MOTP belong to the CLEARMOT metrics family.

HOTA A recommended tracking metric is HOTA which explicitly measures both types of errors related to detection and association and combines MOTA and IDF1 in a balanced way. HOTA also incorporates measurement of the localisation accuracy of tracking results, which is not provided in MOTA or IDF1. HOTA can be thought of as a combination of three IoU scores. It divides the task of evaluating tracking into three main steps: detection, association, and localisation. It calculates a score for each of them using an IoU formulation and then combines these three IoU scores for each of the three steps into the final HOTA score.

2.4 Limitations

When pedestrians are the subjects to be identified and tracked using the computer vision techniques presented, potential issues arise. Pedestrians are often in a small scale and sometimes there are situations of crowdedness: these aspects make computer vision tasks very challenging.

As mentioned in (Wu et al., 2020a) in recent years advances in CNNs and object detection have noticeably improved the performance of pedestrian detection. However, it must be considered that these performances occur in cases where there are large-scale pedestrians, whereas in the case of small-scale pedestrians, a significant deterioration is evidenced due to low resolution implying a lack of detailed information.

This scarcity of information that characterises small pedestrians comes from the fact that the outlines of instances of this type are not as clear as in the case of large pedestrians. Indeed, they often have blurred boundaries and obscure appearances that make detection difficult. This complexity is also due to the small difference between these small pedestrians and the background. This can lead to confusion with anthropomorphic negative samples which are pedestrian-like objects such as vertical structures, dustbins, traffic lights, and trees as mentioned in (Zhao et al., 2020). The shape of these objects is very similar to that of small pedestrians and this leads to incorrect detections.

In the survey (Liu et al., 2021) an attempt is made to summarise and highlight the main problems dealing with small objects:

- Individual feature layers do not contain sufficient information for small object detection.
- Limited context information of small objects.
- Class imbalance for small objects.
- Insufficient positive examples for small objects.

They also propose some techniques in order to improve the performance in detecting small objects such as improving feature maps for small objects, incorporating context information of small objects, correcting foreground and background class imbalance for small objects, and increasing training examples for small objects.

(Cheng et al., 2022) proposes a comprehensive review of small object detection. In this work, the methods used to deal with the limitations presented above are grouped into the following categories: sample-oriented methods, scale-aware methods, attention-based methods feature-imitation methods, context-modelling methods, and focus-and-detect methods.

Since small-scale pedestrian detection is still an open problem many attempts have been made by the researchers such as: (Kim et al., 2021), (Pang et al., 2019), (Yu et al., 2020), (Li et al., 2017), (Akyon et al., 2022).

As with object detection, with object tracking there are some potential issues. The main problems are occlusion which occurs when an object that is being tracked is hidden by another object, and background clutter which occurs when the background near the object is the same colour or texture as the object and this makes the tracking difficult. Other issues can be illumination and shadows, complex object behaviour, and low resolution.

A different but strictly related issue is represented by dense crowds, situations in which there are a lot of people and it is extremely difficult to identify a single person. Crowd analysis and in particular dense crowd scenarios arise as in the previous cases many challenges that make object detection and tracking tasks difficult. Some of these issues for example are occlusions, non-uniform distribution of people, scale, and perspective as mentioned in (Sindagi and

Patel, 2018).

A final consideration concerns ethical issues. There are often difficulties in finding available datasets with the required features due to privacy restrictions on video recording public spaces and people. On the other hand, using simulations to recreate realistic scenarios is often difficult and time expensive. These aspects limit the research in this field.

Chapter 3

Methodology

This thesis work makes use of a tracking-by-detection algorithm based on pedestrian subjects with the aim to discover patterns in urban mobility and characterise pedestrians. This is done to better understand some of the urban and mobility aspects. This is a data-driven and unsupervised approach since the aim is to identify interesting mobility patterns through data provided without any sort of annotation.

The Chapter explains the methodology used to implement the models and the configurations required for pedestrian detection and localisation task. It describes the models and datasets used, different parameter settings and configurations for experimental setup for model training and testing, the metrics used for evaluating the model performance and the analyses carried out related to urban concepts.

3.1 Overview

The initial phase of the project comprised a literature review of previous research on existing methods for pedestrian detection and tracking, highlighting the main limitations. YOLOv7 and SORT were the algorithms chosen for object detection and object tracking step respectively. These two algorithms were chosen considering ease of use, open source accessibility, and speed.

The following phase involved the research of a suitable dataset for the training phase for object detection algorithm. The Centre de Gestion de la Mobilité Urbaine (CGMU) dataset was chosen for YOLO object detector model training. At the same time, a review of existing datasets for training object trackers was performed pointing out the scarcity and complexity of available datasets which fits specific requirements (large-scale public space, small pedestrians, fixed webcam videos).

Once the training phase of the selected object detection algorithm was completed, detection and tracking were applied to the videos of Piazza Duomo using the trained weights. Before conducting urban analysis and exploring

spatial utilisation it was necessary to reproject on the plane the detections obtained through a georeferencing process.

In order to perform urban analysis QGIS software and Python scripts were used. More specifically, the georeferenced data were considered to initially analyse the distribution of the detections using e.g. heatmaps. Afterwards, considering the following detections in the form of trajectories, the potential users' profiles, commuters and tourists, were analysed, taking into consideration some main characteristics such as speed, duration, distance travelled and direction. Finally, the presence or absence of groups in the area considered was investigated. To support this analysis, t-tests were also performed.

Lastly, an analysis and discussion of the results obtained have been developed.

The framework for the project can be seen in Figure 3.1.

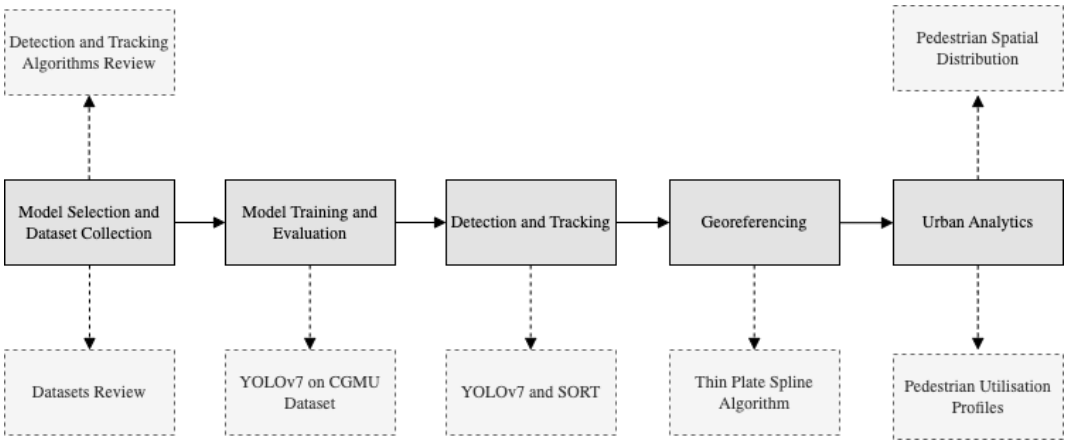


Figure 3.1: Overview of proposed framework

3.2 Experimental Setup

3.2.1 Computational Resources

In order to execute Python scripts in a more efficient way, specifically those relating to the training phase of the object detection model, it was necessary to use Google Colab Pro+ [1]. It allows users to write and run Python codes through browser. Google Colab is becoming more and more popular in developing deep learning solutions. It is a hosted Jupyter notebook service which provides free access to computational resources based on cloud. Another advantage is the possibility to connect to Google Drive and use the files there without having to upload and download them at every session.

There are different subscription plans for this service. The basic version offers free resources, although resources are not guaranteed and are not unlimited.

[1] https://colab.research.google.com/

For this reason, it is possible to use the paid plans: "Pay As You Go", "Colab Pro", and "Colab Pro+". "Google Colab Pro+" variant compared to the free and Pro plan offers even more options to run deep learning relatively inexpensively without a cloud server or local machine. In particular, Colab Pro+ offers background execution which supports continuous code execution for up to 24 hours, 52 GB RAM memory available with the high-memory option, and fast GPUs like the T4 and P100 if resources are available.

3.2.2 Specification of Dependencies

Python (version *3.8.10*) is the programming language for the project used in the Google Colab platform. Python is a very popular open-source computer programming language that is being used in different disciplines. It has an object-oriented approach that facilitates coding in a logical and clear way.

Different Python libraries were used for developing the project. In particular, GeoPandas (*0.12.2*) was used for the analysis of geospatial data, which enables the use of Pandas functionalities with geospatial data structures as well. For the image processing tasks, OpenCV (*4.6.0*) was used. For the analysis of detections and trajectories, MovingPandas (*0.15.rc1*), PyMove (*3.0.1*), and pointpats (*2.2.0*) libraries were used. Folium (*0.12.1.post1*), contextily (*1.3.0*), matplotlib (*3.2.2*) and seaborn (*0.11.2*) were used for the generation of visualisations.

3.2.3 Other Tools

QGIS[2] (*3.22.13*) has been utilised to perform the georeferencing step and the first stage related to urban analytics. QGIS is a free and open-source under the GNU General Public License cross-platform desktop geographic information system application that supports viewing, editing, printing, and analysis of geospatial data, founded by Gary Sherman in 2002. Other open-source GIS software are GRASS, OpenJump, and uDig. There are also proprietary GIS software such as ArcGIS, the most common proprietary GIS program, MapInfo, GeoMedia, GeoBas, DP/Map, and Global Mapper.

Other tools utilised are Roboflow [3] (*0.2.31*), at an early stage to set up the dataset, Weight and Biases [4] (*0.13.10*) to monitor the training phase.

3.3 Model Selection

Following the literature study, the assessment of existing methods and their characteristics was performed in order to gain knowledge and deep understanding of different methods that can be used for such purposes, identifying the most suitable ones.

[2]https://qgis.org/en/site/
[3]https://roboflow.com/
[4]https://wandb.ai/site

For object detection, the YOLO family of models presented in 2.2.3 was selected as the most promising. YOLO was chosen considering the scope of this study for its speed and accuracy in real-time as well as the public availability of its source code compared to other object detection methods. YOLO provides also a better generalisation for new domain. The models that were specifically considered were YOLOv5 and YOLOv7. The following two models were trained on the same dataset with the same parameters configuration in order to identify which of the two performed better.

Considering object tracking, SORT family of models was considered for its real-time speed and accuracy amongst competing state-of-the-art models. As seen in 2.3.3 there are different implementations: SORT, deepSORT, and strongSORT. In particular, the first one, SORT, and the last one based on deep learning techniques, strongSORT, were the models taken into consideration given the ease of their implementation. Both models were tested in combination with YOLO to determine which one was better.

3.4 Dataset Collection

Before training the model, it was necessary to identify a suitable dataset. The performance of machine learning and deep learning models depends heavily on the quality of the data on which they operate. Therefore, the training phase of a model is of fundamental importance, a process by which the model learns to identify particular features from labelled data and which allows it to improve its performance.

Two different datasets were used for this work. One of these was used for the training phase of the object detection model, YOLO, while the other dataset was the one used to perform the urban planning analysis. In particular, the dataset on which computer vision techniques were applied and validated is the dataset containing videos of Piazza Duomo.

Datasets required for training detection algorithms need to satisfy certain characteristics in order to have a scenario similar to the final one considered for analysis: small-scale pedestrians, stationary cameras, large-scale public spaces, and similar angle of view. In addition, in order to succeed in the training steps, datasets need to be already annotated. In fact, the annotation phase is a long and laborious process that required a collective effort and specialized software.

At an early stage, a detailed analysis of the most used datasets for object detection task regarding people such as Caltech, INRIA, CityPersons, EuroCity Persons, CrowdHuman and WiderPedestrian Challenge was carried out. However none of those satisfied the requirements previously introduced. In some cases, these were videos containing large-scale pedestrians or videos taken in motion. CGMU dataset (City of Montreal, 2020), described in 3.4.1, represented a good solution. This dataset satisfies the previously mentioned requirements and has already been used in some similar projects such as in (Messa

et al., 2022). By looking at a few frames of the available videos it is possible to observe the presence of small to medium-sized pedestrians and high angle perspective camera.

3.4.1 CGMU Dataset

The object detection model was trained on CGMU dataset. The training phase was useful for improving the performance of the object detector.

The CGMU dataset is obtained from videos captured by 500 Pan-Tilt-Zoom (PTZ) traffic cameras installed in the city of Montreal. These PTZ traffic cameras are used by Montreal's Centre de Gestion de la Mobilité Urbaine for traffic management and therefore to manage the city's mobility, on a daily basis. It is composed of 10,002 images (see Figure 3.2) of different resolutions 350x288, 352x240, and 704x480 with their associated object detection annotations. The images that constitute the dataset were captured between May and July 2019. The date and the camera identifiers are indicated on the video frames.

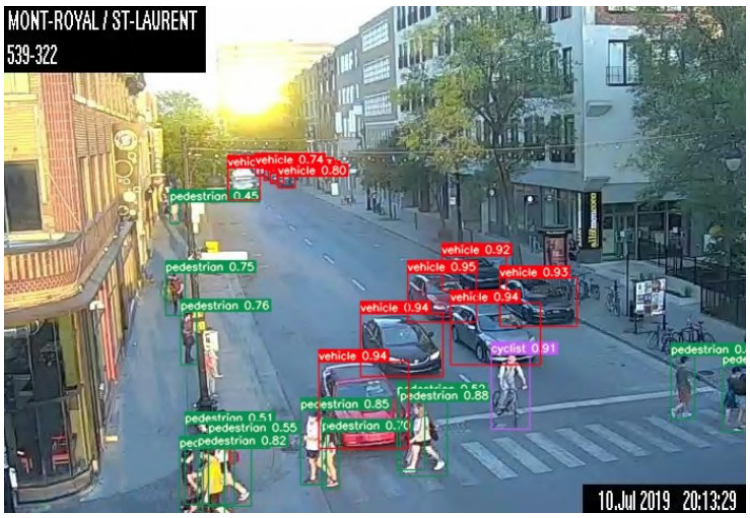
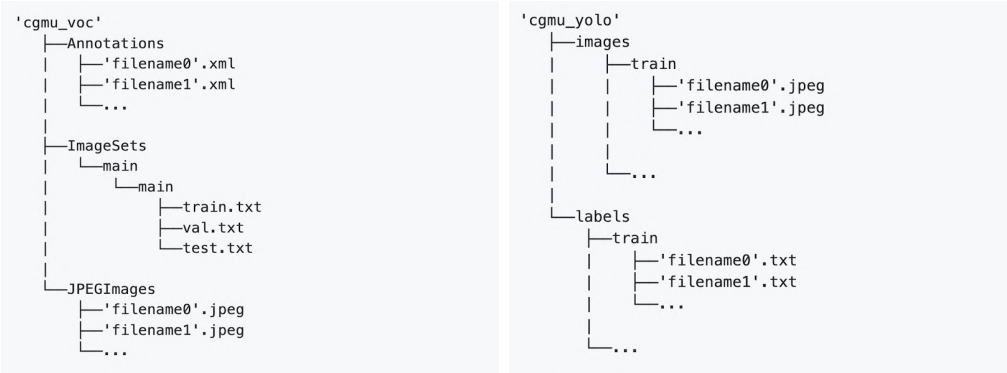


Figure 3.2: Frame of CGMU dataset

These images will be used to train the algorithm to detect pedestrians, hence they are associated with annotations containing some information about the identified object, i.e. the class of the identified object and the coordinates of the bounding box. The five classes of objects that can be identified are vehicles, pedestrians, construction objects, buses, and cyclists. The cyclist class only includes a person on a bicycle (or scooter/ unicycle/ electric bicycle), so it excludes single bicycles and people walking alongside their bicycle. All other motorised modes of transport are instead considered as vehicles or buses. The setting of the cameras used for the analysis makes it possible to respect the privacy of the people identified, as it is not possible to capture the faces nor the license plates of cars. Furthermore, it is not possible to accurately track the movements of people since the images are collected in 5-minute intervals. Some of the images do not have any annotation and this means that no objects of the

above classes mentioned before are present. A null annotation is different from a missing annotation. The latter occurs when an image has objects but they are not annotated. This could be problematic as your model will be trained on false negatives of your objects while a null annotation is not necessarily problematic in fact, it may be desired in order to train a model that objects are not always present in the frame. In this dataset, there are no missing annotations.

The annotations were generated using a Computer Vision Annotation Tool (CVAT) and they are represented by XML files. The archive is in Pascal VOC format (see Figure 3.3a).



(a) VOC format (b) YOLO format

Figure 3.3: Annotation formats

This format includes three main directories:

- JPEGImages: source images from traffic cameras (.jpeg).
- Annotations: annotations (.xml) related to the images in the dataset, including the following attributes: filename, size of the image, information about the objects identified in the image (label indicating the class of the objects and bounding boxes coordinates).
- ImageSets: different .txt files which refer to the composition of the training, validation, and test sets.

To train YOLOv5 model on CGMU dataset, it must first be converted into a format that is compatible with YOLO (see Figure 3.3b), with a structure consisting of two folders, images and labels, and within these the corresponding images and annotations. In order to convert the CGMU dataset from Pascal VOC to the YOLO format a specific Python script has been used.

The dataset was then loaded on Roboflow where it can be inspected deeply. In addition as preprocessing step a resize operation was performed. The images have been resized to 704x704 pixel size, a standard size accepted by YOLO in the training phase. The resulting dataset contains 9,995 images since 7 images have been identified as corrupted. Later, the uploaded dataset was split into training, validation, and test set in a proportional way. 70% of the dataset

used for training, 20% for validating the results of the trained model, and 10% for testing it. In particular, there are 6,996 images in train set, 1,999 images in validation set, and 1,000 images in test set. The table below, Table 3.1, shows how many instances of each class are present.

	Number of Objects Seen in Dataset					
	vehicle	*pedestrian*	*construction*	*cyclist*	*bus*	*total*
Train	64,423	10,781	8,839	837	1,019	85,899
Validation	18,390	3,304	2,637	221	281	24,833
Test	8,917	1,638	1,287	125	144	12,111
Total	91,730	15,723	12,763	1,183	1,444	122,843

Table 3.1: CGMU Train, Validation, and Test set

The distribution of classes is unbalanced, with far more vehicles than others classes. The total number of instances in the dataset is 122,844. In training, validation and test set the percentages related to the presence of different classes of objects are constant. In fact, the percentage of vehicles in the three splits of the dataset is about 74.0% of the total annotations, the percentage of pedestrians about 12.5%, the percentage of construction about 10.5%, the percentage of cyclists about 1.0%, and the percentage of bus about 1.2%.

3.4.2 Piazza Duomo Dataset

The dataset used to perform the analysis is the webcam video dataset of Piazza Duomo (see Figure 3.4), one of the most important squares in Italy characterised by an important flow of people due to its historical importance and strategic position.

Figure 3.4: Frame of Piazza Duomo dataset

The decision to focus on this square is based on the fact that the square considered can be used in different ways and by different pedestrian profiles. Indeed, it is crossed by people on their way to work but also by many tourists given the presence of shops and attractions in the surrounding area. Moreover,

another relevant element characterising the square is the presence of subway entrances.

The Piazza Duomo dataset consists of videos recorded by stationary cameras installed in the respective squares. Videos are captured on Thursday, July 15, 2021, in different 30-minute time slots, five different moments of the day: 8:00 - 8:30, 11:00 - 11:30, 12:45 - 13:15, 15:00 - 15:30, and 18:00 - 18:30. The total duration of the videos available is therefore 5 hours. Videos have 1920×1080 pixels resolution and a frame rate equal to 15.01 FPS (Frames Per Second). These videos do not contain annotated data differently from the CGMU dataset which is used for the training phase.

The framing is fixed and it is at the top of one of the buildings facing the central part of the square. A portion of the square from the video is covered by the presence of the central equestrian statue which obstructs the view. The main subjects of these datasets are small-scale pedestrians. In addition to pedestrians, cyclists, and vehicles with special permits can be spotted infrequently. Furthermore, there are some attractors and shaded areas. The area of the square considered for the analyses is approximately 12,000 square meters. The main characteristics of the squares are reported in Figure 3.5. The figure shows the entrances to the subway, the location of the equestrian statue and the position of the CCTV camera.

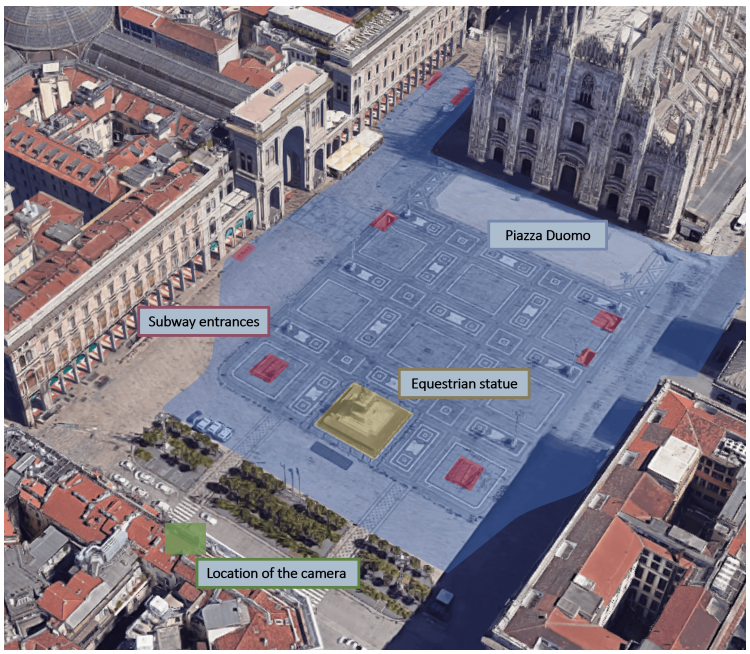


Figure 3.5: Piazza Duomo study area

Performing a high-level manual count, it can be assumed that there are approximately 300 people per frame. Obviously, this number is highly variable depending on the time slot considered. The pedestrians identified are located rather far away from the camera, their size is about 30 pixels.

3.5 Model Training and Evaluation

Training on CGMU dataset described in paragraph 3.4.1 was performed for both object detection models, YOLOv5 and YOLOv7. Model training as well as testing implementation will be conducted in the Google Colab environment.

When training a neural network several parameters can be tuned to give better performance. The parameters considered and their values are the same for YOLOv5 and YOLOv7. The ones evaluated in this thesis are:

- *batch-size*: When training data is large, providing the entire dataset to the model implies a very large number of weights that the network has to learn. For this reason, the input is divided into batches. The batch-size indicates the number of images in each batch. The larger this value, the more memory consumption will be consumed.
- *weights*: If no path is indicated, the model will consider random weights for training.
- *cfg*: This is the path to the model configuration file which is needed for loading the model architecture. It allows the train.py file to compile and build the architecture specified for training input images. This file contains the number and types of layers, how they are interconnected, and the number of classes in the selected dataset.
- *data*: Specifies the path to the dataset YAML file. In this way, the model will be able to reach the different splits of the dataset: train, validation, and test set. The number of classes to be considered in the detection phase and the name of these is also indicated.
- *img-size*: Defines the input image size, by default, the images will be resized to 640×640 resolution before being fed to the network.
- *epochs*: The number of times to cycle through training data. The number of epochs indicates how frequently the model learns all its inputs and changes its weights consequently to get closer to the ground truth labels.
- *device*: The GPU number (ID) to use for training. In case of one GPU this value is 0.
- *hyp*: Different set of parameters and hyperparameters for training.

According to the computational capacity of the available GPU, the *batch-size* value was set to 32.

The transfer-learned model uses the weights of a model pre-trained on the Microsoft Common Objects in Context (MS-COCO) dataset as a starting point and then fine-tunes only the output layers for detection on more specific objects. In this case yolov7x_training.pt and yolov5x.pt were chosen since they are created for transfer learning purpose. YOLO contains different network

structures due to different widths and depths: YOLOv7x and YOLOv5x are the largest considering 640 as the test size.

The cfg file which describes the layout of the network, block by block, corresponds to the cfg file related to YOLOv7x and YOLOv5x models. No further changes have been made.

The data.yaml file indicates the path of train, validation and test set. It also specifies the number of classes considered. In this case, considering CGMU dataset, there are 5 classes, respectively bus, construction, cyclist, pedestrian, and vehicle.

Since the final image size set in Roboflow was 704*704 *img-size* parameter value was set to that value.

The experiments were performed for 100 epochs for each of the two models. This value has been chosen after experimenting with different epoch values and observing the loss, accuracy, and performance results. Training the model for a specific number of epochs allows to avoid over-fitting as well as increase the generalisation capacity.

Since Google Colab Pro+ GPU is utilised the *device* parameter is set to 0.

The file related to hyperparameters, hyp.scratch.p5, is the default configuration spec.

The training process was monitored in real-time through Weight and Biases.

Once the training step of the object detector model on CGMU dataset was completed, the performance evaluation on validation and test set was carried out. The metrics considered are those described in Chapter 2.2.4. As for training, there are some important parameters to set. Some of these correspond to the ones presented for the training process. Considering the testing step, the parameter *task* has to be set to 'test' in order to indicate the correct path for test data. The parameter weights indicate the best weights obtained in the previous phase of training which will be used in this evaluation phase. The other two important parameters are:

- *conf-thres*: It indicates the confidence threshold. It is the minimum score that the model will consider the prediction to be a true prediction otherwise it will ignore this prediction entirely.
- *iou-thres*: It indicates the IoU threshold and it is the minimum overlap between ground truth and prediction boxes for the prediction to be considered a true positive.

The values of the latter two parameters have been varied in order to find the best combination for the desired results.

3.6 Detection and Tracking

Once the training step of the object detection models was completed, in combination with the best selected object detector, two different tracking algorithms, SORT and StrongSORT, described in Chapter 2, were tested. In this case, the choice of the best model was based on speed performance measured in milliseconds. After selecting the best model for tracking, detection and tracking process was carried out on Piazza Duomo dataset.

The result of the object recognition and tracking process is the creation of a text document with the following information: video frame, object ID, object class (i.e., *pedestrian*, *vehicle*), pixel X, and pixel Y.

The X and Y coordinates obtained refer to the coordinates of the top-left corner of the bounding boxes related to the identified objects. This is one of the different formats used by object detectors to indicate the coordinates of bounding boxes. Each format uses its specific representation. The main formats are:

- *COCO format [x_ min, y_ min, width, height]*: the coordinates (x_min, y_min) are the top-left corner along with the width and height of the bounding box.
- *YOLO format [x_ centre, y_ centre, width, height]*: x_centre and y_centre are the normalised coordinates of the centre of the bounding box. The width and height are the normalised length. To convert YOLO in COCO or PASCAL VOC or vice versa it is important to have the size of the image to calculate the normalisation.
- *PASCAL VOC format [x_ min, y_ min, x_ max, y_ max]*: x_min and y_min are coordinates of the top-left corner and x_max and y_max are coordinates of bottom-right corner of the bounding box.
- *albumentations format [x_ min, y_ min, x_ max, y_ max]*: it is similar to PASCAL VOC format but in this case the coordinates are normalised.

In this case, the algorithm gives coordinates in COCO format (see Figure 3.6), [x_min, y_min, width, height].

In order to obtain the location at ground of the pedestrian a preprocessing operation is required to obtain the middle point of the lower side of the bounding box. The coordinates are multiplied by width and height to have not normalised values and then the X value is calculated as *x_ min + (width/2)* and Y values as *y_ min + height*. To do this, it was necessary to modify the *detect_ and_ track.py* script.

3.7 Georeferencing

Unlike other types of data obtained through different detection tools, data obtained from webcam videos cannot be analysed directly but require some

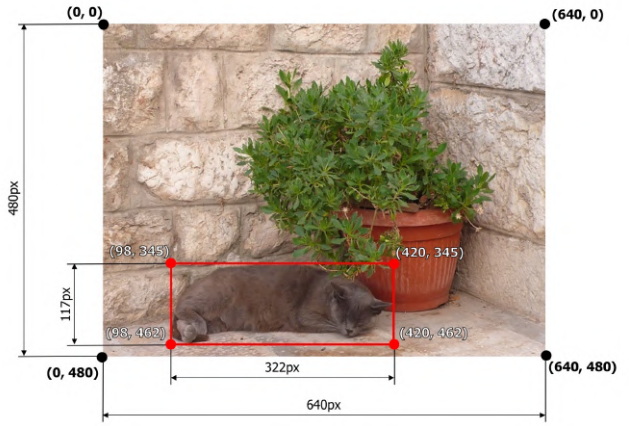


Figure 3.6: An example image with a bounding box from the COCO dataset

operations. Indeed, once the coordinates in the proper format of the identified pedestrians have been obtained, it is necessary to proceed with the georeferencing step using a Python script and QGIS software in order to locate the detections in the target area. The georeferencing step estimates the geographical position of the distribution of pedestrians in the square starting from the previously identified coordinates (i.e., pixel X, pixel Y) and translating them into geographic coordinates. EPSG:32632 is the projected coordinate reference system considered and it is based on the Universal Transverse Mercator (UTM) projection for zone 32N. Coordinates are expressed in meters and they are calculated relative to the origin of the reference system, which lies on the equator line, 500,000 meters east from the reference meridian, which for zone 32N corresponds to 9° east longitude.

To perform georeferencing, it is necessary to use the QGIS implementation of the Thin Plate Spline algorithm as a transformation technique. This technique is based on a series of Ground Control Points selected to anchor points in the perspective image through a continuous function. Thin Plate Spline algorithm models the complex non-linear spatial relationship between the ground control points and their corresponding locations in the projected coordinate system. The Ground Control Points in this situation represent specific features of the area that is being analysed and can be streetlamps, corners of buildings or coloured cobblestone patterns. The more points there are, the better the results will be. In the case examined, 354 GCPs were identified (see Figure 3.7), most of them relating to the geometric patterns that characterise the marble and granite paving of the square.

More specifically on how it works, in QGIS a random video frame from Piazza Duomo is loaded. On this frame, using Georeferencer QGIS plugin, some ground control points (GCPs) are identified (the more the better) on the frame and then located on the map. In this way through Thin Plate Spline algorithm the frame is georeferenced (see Figure 3.8). This is done iteratively through a Python script for all frames that constitute the videos.

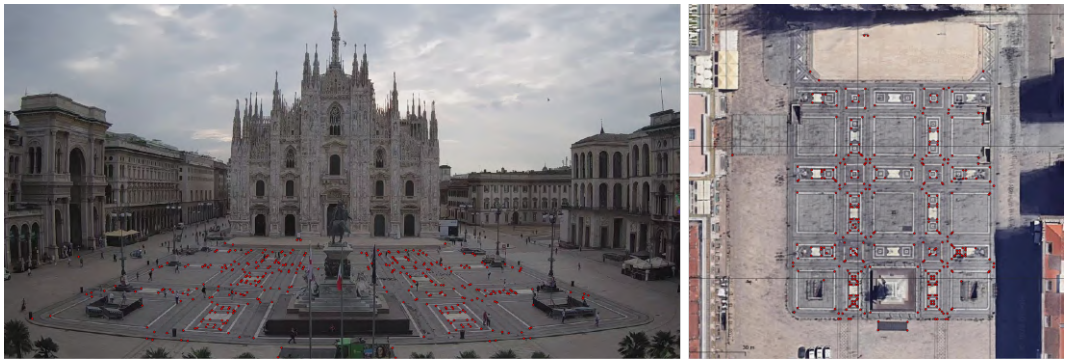


Figure 3.7: Piazza Duomo Ground Control Points

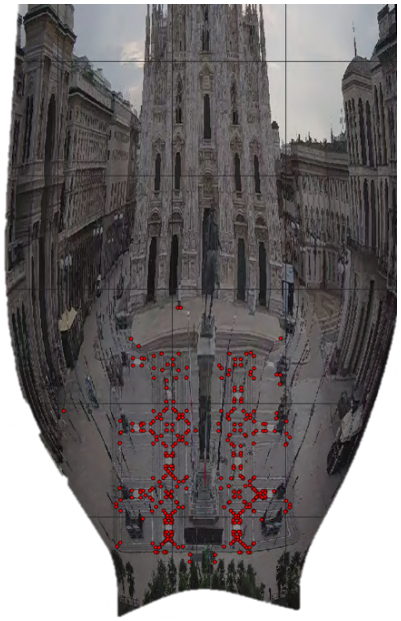


Figure 3.8: Georefrerencing Thin Plate Spline algorithm results

Since these videos have a complicated perspective some points could not have their corresponding in the georeferenced frame. The main risk is that some detections could not be georeferenced. In order to find a solution, these ground control points are loaded on an image of random unique pixel with the same dimensions of the previous frame. In this way there is a unique correspondence for the pixel of the two images and in a similar way the image with random pixel is georeferenced. This avoids the case that some points where there are identified pedestrians are lost.

Using Python scripts a GeoJSON file was generated after linking all the detection in the videos to the real coordinates. The resulting geodataframe is containing all the detection points georeferenced. Importing this file in QGIS it is possible to visualise these georeferenced detections on map with real-world coordinates.

3.8 Urban Analytics

This part of the analysis is focused on the production of maps and graphics obtained through the open-source software QGIS and Python scripts based on the georeferenced data obtained. The aim is to investigate the presence of patterns of usage in the area considered. In particular, the distribution of pedestrian detections in the square and their trajectories are analysed. Based on these, an attempt is made to identify pedestrian profiles and the presence or absence of groups of people crossing the square.

3.8.1 GIS Analysis and Mapping

QGIS was used to display all the detections for the different time windows on the map. This process made it possible to verify the accuracy of the georeferencing step of the detections originally obtained. Through the use of QGIS, the distribution of pedestrians in the square was then analysed for high-level patterns. Furthermore, using the preprocessing tools of QGIS, a heatmap was generated, which provided a general idea of the areas of the square characterised by a higher density.

3.8.2 Point Pattern Analysis

After using QGIS, Python libraries were used to perform more detailed analyses. In particular, two notebooks were created using Python for point pattern analysis and trajectory data mining, described in Chapter 2.

More specifically, the following analyses will be carried out:

- *Distribution map, Descriptive statistics*: Map and synthetic values allowing high-level analysis of the distribution of pedestrians in the square.
- *Quadrat analysis*: Analysis of the pedestrians' distribution into equally-sized quadrats to understand high-level patterns in the area considered.
- *Heatmap, KDE heatmap*: Analysis of the pedestrians' distribution by counting the number of detections in a continuous way, not by discretising the area.
- S*tandard metrics calculation (Occupancy, Density, Flow Rate)*: Calculation of urban planning metrics to understand the effective use of the square.

First detections considered as outliers according to X and Y values were discarded. The number of outliers identified is very limited and it is due to errors in the detection phase. Then, graphs were generated to display the georeferenced detections as a whole, obtaining results similar to those obtained previously with QGIS with the aim to understand the areas with a higher density of detections.

In order to conduct more in-depth analyses, a density-based analysis was performed. In particular, a quadrat analysis was implemented, which allows to divide the study area into sub-regions of similar shapes (in this case quadrats) and to count the number of detections in each of them. This kind of analysis can be helpful in investigating the spatial distribution of points and the presence of patterns.

However, this approach is limited by the choice of areas used for the counts. This aspect leads to Modifiable Areal Unit Problem (MAUP), a statistical bias that arises by discretising an area and aggregating point data. In fact, the areas used for analysis are often chosen arbitrarily by the analyst, and with different choices different results could be obtained. To overcome this problem heatmaps and Kernel density estimation (KDE, an empirical approximation of the probability density function) heatmaps which allow counting the number of points in a continuous way were generated. The result obtained is more homogeneous and provides a better generalisation of the theoretical probability distribution over space while still capturing the same structure as the previously proposed analyses.

To support this kind of analysis, some metrics were calculated, such as the average *Occupancy (pedestrian/cell)*, the average pedestrian *Flow Rate (pedestrian/minute/meter)*, and the average level of *Density (pedestrian/squared meter)*. The *Occupancy* metric is calculated by measuring the number of pedestrians in the square considering a grid of the analysed area consisting of 4 square meter cells (see Figure 3.9).

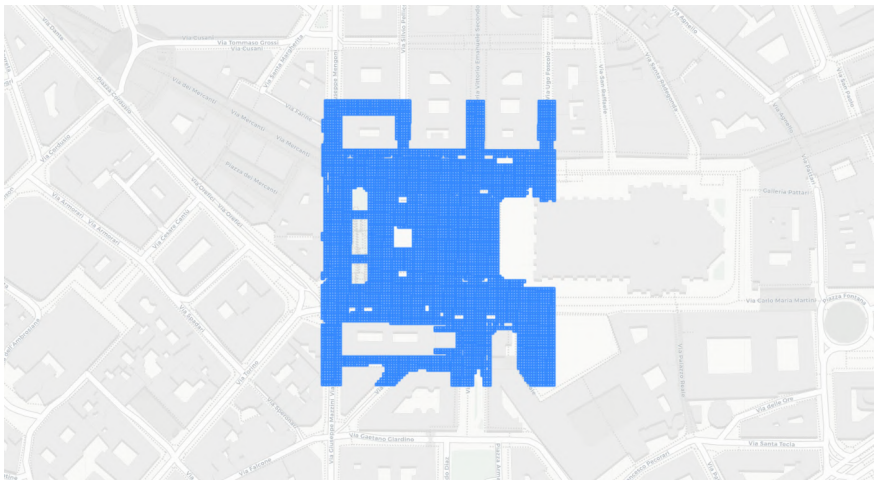


Figure 3.9: 2m x 2m Piazza Duomo grid

In particular, for each grid cell, the number of unique pedestrians is measured. The average pedestrian *Density* is obtained by counting the number of people who occupied the considered area considering each frame of the videos and then calculating average values for each cell. The average level of *Flow Rate* counts the number of pedestrians walking through the delimited area at one-minute intervals.

These two latter results can be interpreted by considering Walkway Level of Service Criteria (LOS) (see Table 3.2), a set of values describing the impact of contextual situations of density on pedestrian circulation dynamics (from LOS A-pedestrian free flow to LOS F-extremely difficulty in walking movements).

Level of Service Criteria	Level of Density (pedestrian/squared meter)	Pedestrian Flow Rate (pedestrian/minute/meter)
A - free flow	≤ 0.08	≤ 7
B - minor conflicts	0.08 - 0.27	7 - 23
C - slower speed	0.27 - 0.45	23 - 33
D - restricted most	0.45 - 0.69	33 - 49
E - restricted all	0.69 - 1.66	49 - 82
F - shuffling	≥ 1.66	≥ 82

Table 3.2: The Walkway Level of Service criteria (Fruin, 1971; Gorrini et al., 2016).

The first LOS values refer to environments in which pedestrians can walk without encountering obstacles while maintaining an adequate distance between them. From Level C onwards, obstacles begin to have a significant impact on people's mobility, up to the last level where movements are completely restricted.

3.8.3 Trajectory Data Mining

The first part of these analyses involved a detailed exploration of the distribution of pedestrian detections, considered as points. On the other hand, these points when considered in aggregate form can be seen as part of trajectories. The purpose of this part of the analysis focuses on analysing the trajectories of pedestrians identified during the frames.

In this case, the following analyses will include:

- *Trajectory distribution map*: Map which provides an overview of the main features of the trajectories from a visual point of view.
- *Trajectory clustering*: Identification and analysis of potential pedestrian clusters based on trajectory patterns.
- *Groups detection*: Identification and analysis of single pedestrians and groups of pedestrians' trajectories.

Before performing the trajectory analyses, however, it was necessary to perform preprocessing operations in order to obtain meaningful and high-quality trajectories. To do this, a workflow consisting of several steps was designed.

Considering the given video frame-rate of 15.01 frames/second, all frames whose number is a multiple of 15 were selected for later analysis. This allowed to consider frames at regular intervals of approximately 1 second. This choice made it possible to reduce the computational complexity required for

the analyses conducted and to have greater convenience in the calculations performed. In addition, only trajectories whose duration is longer than 20 seconds were considered. This threshold was selected considering a hypothetical minimum route travelled by a person in the square considered. The route to the centre of the square from one of the subway entrances (about 30 m) is considered as the minimum route, in fact assuming a constant speed of 1.5 m/s it takes 20 seconds.

Before starting data mining tasks it was necessary to identify and remove trajectories containing outliers (see Figure 3.10) in order to obtain a subset of accurate trajectories.

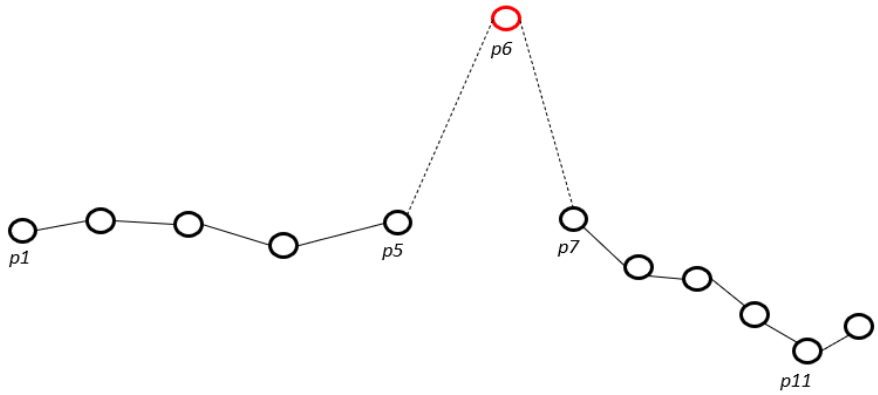


Figure 3.10: Outlier point in a trajectory

A heuristics-based approach was applied considering the travel speed of each point of the trajectory. Speed dimension was added for each point part of the trajectories using MovingPandas, a Python library. A threshold has been set so that trajectories containing points that have a speed less than this value are discarded. The threshold initially considered was set to 1.5 m/s according to (Crociani et al., 2019; Vanumu et al., 2017). However, carrying out some tests and observing the spatial and more quantitative results, it emerged that with this value, a large percentage of trajectories were eliminated, especially the longer ones. Therefore, considering this and the limited accuracy of the algorithms used, it was decided to use a slightly higher threshold, 2.0 m/s.

Finally, by aggregating the data of the above preprocessed trajectories, some thematic attributes can be derived from trajectory data as described in (Andrienko et al., 2013). The following features were generated using MovingPandas library: average speed, average direction, travelled distance, and duration in time. The direction, whose values are in degrees, starting North turning clockwise, is calculated between consecutive locations and the average value for the entire trajectory is then calculated. This features generation step allows the transition from raw trajectories to semantic trajectories. Using these data, the trajectories were analysed and tried to cluster them based on similar

characteristics. In particular trajectory clustering was performed with the aim to identify two specific clusters of pedestrians, commuters and tourists. Various algorithms were considered to perform clustering operation. Algorithms belonging to the three different families presented in Chapter 2 were tested: partition-based, hierarchical-based, and density-based. In particular, K-Means was implemented for partition-based approaches, AgglomerativeClustering and Birch algorithms for hierarchical-based ones, and DBSCAN and OPTICS for density-based approaches. Finally, considering the obtained results, K-Means was chosen. Since K-Means is sensitive to the scale of the variables in the dataset, it was necessary to standardise the features considered before proceeding. The K-Means algorithm partitions the data into k mutually exclusive clusters and provides as output a vector of indices indicating to which of the k clusters each observation belongs. Considering the two clusters obtained, several features such as numerosity, speed, distance travelled, and duration were analysed with the aim of interpreting the clusters and finding out the main properties of the two categories of pedestrians considered, commuters and tourists. In order to further investigate the differences between the two types of pedestrians, t-tests were carried out considering the different time periods and the various metrics calculated.

As an auxiliary analysis, the presence or absence of groups was investigated. A group can be considered as a set of detections spaced less than a certain threshold for the same interval of time. The threshold chosen is 1.5 m (Crociani et al., 2019). Another assumption made was to consider only groups of trajectories whose start and end points are close enough, no more than 4 m. It means that two or more pedestrians in order to be considered part of a group should not have diverging trajectories. Out of the potential groups identified, only those consisting of two or three elements were considered. The main characteristics were analysed further on and lastly, as in the previous case, t-tests were performed to prove the differences between single pedestrians and groups.

Chapter 4

Results and Discussion

In the following Chapter there's a presentation and analysis of the results obtained implementing the methodology shown in Chapter 3. The results of the training phase of the object detection model and the results obtained on the videos of Piazza Duomo are presented. Furthermore, findings from the clustering and groups detection phase are shown through the analysis of metrics and the use of graphs and maps.

4.1 YOLO Results

Considering the two models selected for object detection, YOLOv5 and YOLOv7, a first training step is presented in order to improve the accuracy of the detectors. As it was stated in Chapter 2, during training the goal is to minimise the loss function. Once the training step has been completed a validation of the performances was carried out on the validation set at first and then on the test set.

4.1.1 Train and Validation Results

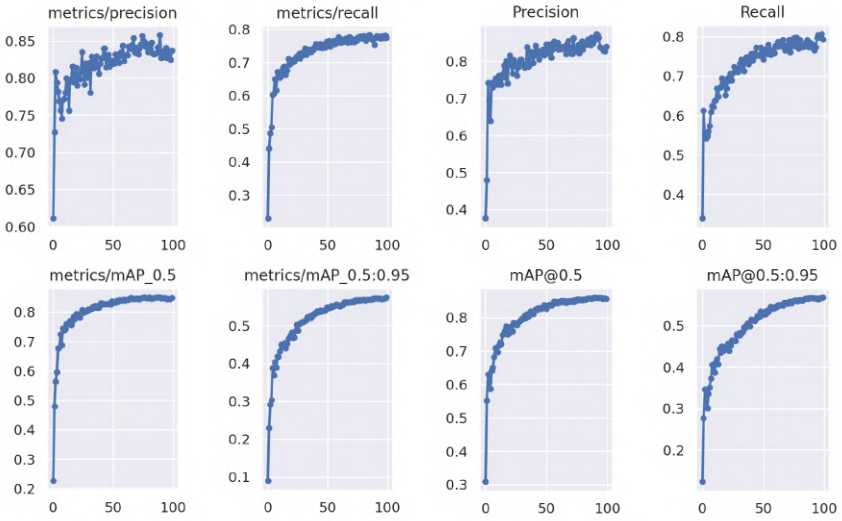
As described in 3.5, YOLOv5 and YOLOv7 training was performed using identical parameters in order to have a robust comparison.

The total execution time for 100 epochs was 4h 17m 2s and 5h 41m 4s respectively for YOLOv5 and YOLOv7.

Analysing the training results and in particular the performances of metrics *mAP_0.5*, *precision*, and *recall* on validation set, it can be seen that YOLOv5 and YOLOv7 achieve the same *precision* value, 0.839. Regarding *mAP_0.5* and *recall*, YOLOv7 had higher values, 0.857 and 0.792 respectively, in contrast to YOLOv5's values of 0.848 and 0.839.

In order to have a better understanding of the performance of metrics and loss functions during the training and validation phases graphs generated by WandB were analysed. For both models, YOLOv5 and YOLOv7, the loss

metrics over the epochs decrease while metrics like precision, recall, and *mAP* increase as expected. The fact that the loss metrics decrease over the epochs indicates that the models considered generalise well on new datasets. In the following graphs, 4.1a and 4.1b, is shown the behaviour of the main metrics (precision, recall, *mAP_ 0.5*, and *mAP_ 0.5:0.95*) and as can be seen, the metrics improve as the epochs progress.



(a) YOLOv5 (b) YOLOv7

Figure 4.1: Validation set performance metrics

The last step consists in saving the best weights for YOLOv5 and YOLOv7 generated from the training process.

4.1.2 Testing Results

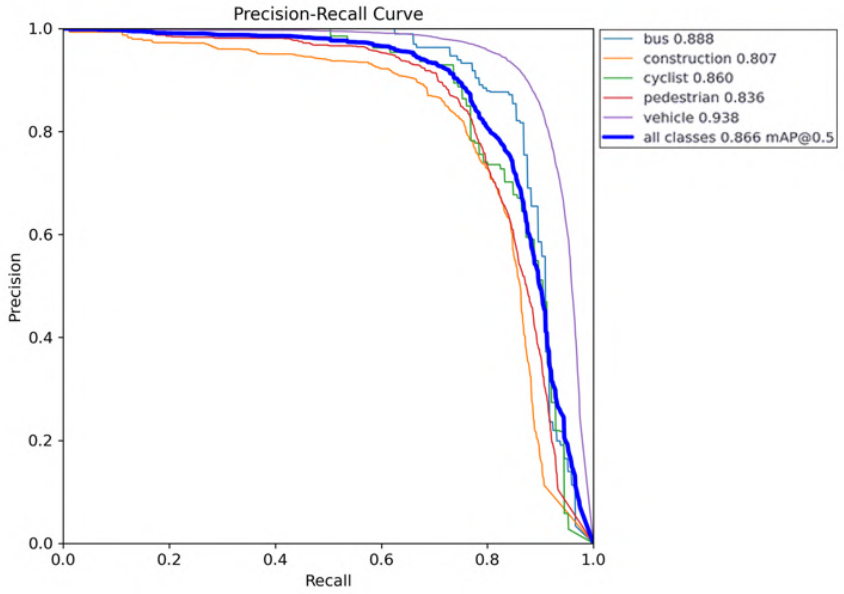
The testing process was necessary to assess the quality of the trained models. Using the best weights obtained in the previous training step, the performance of the models on the test set were evaluated. The parameters are analogous to those presented for the training process and are described in 3.5. For the values of *conf-thres* and *iou-thres*, 0.001 and 0.65, YOLOv7 default values, were used.

YOLOv7, as on the validation set, performed better on the test set as well, with *mAP_ 0.5* of 0.874, *precision* score of 0.860, and *recall* of 0.799 (see Table 4.1).

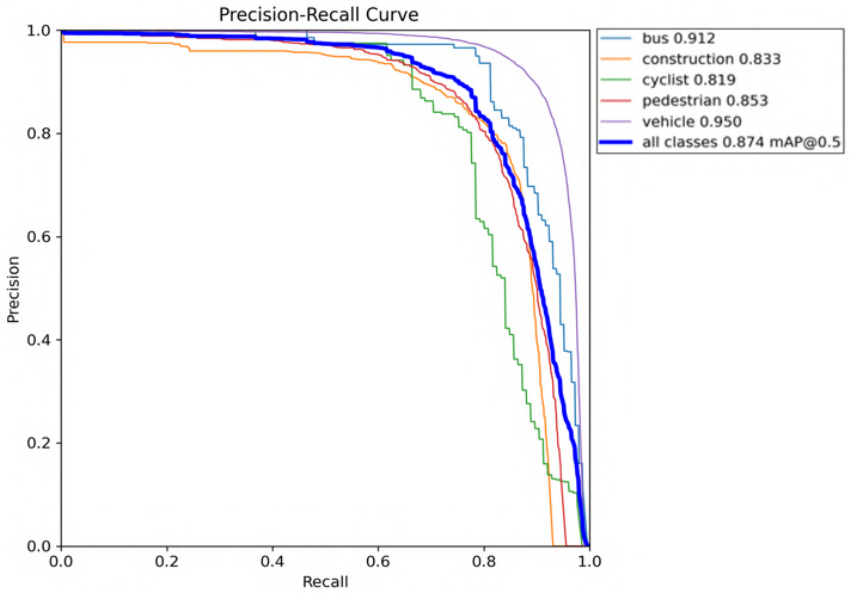
YOLOv5			**YOLOv7**		
mAP_ 0.5	*precision*	*recall*	*mAP_ 0.5*	*precision*	*recall*
0.866	0.845	0.798	0.874	0.860	0.799

Table 4.1: Performances on test set on all classes

With Figure 4.2, it is possible to analyse the performance on the different classes of the identified objects. More specifically, for the "pedestrian" class, the one studied, *mAP_0.5* has a value of 0.836 for YOLOv5 and 0.853 for YOLOv7.



(a) YOLOv5



(b) YOLOv7

Figure 4.2: Testing set precision-recall curves

In conclusion, it can be stated that YOLOv7 trained using the same parameters as YOLOv5 achieves better results.

4.2 Inferencing on Piazza Duomo

Once the training and validation step of the models was completed, YOLOv7, which was proven to perform better than YOLOv5, was chosen for the following stages. In order to demonstrate the effectiveness of the training process, YOLOv7 was tested on an extrapolated frame from one of the videos of Piazza Duomo with pre-trained weights on a general-purpose dataset, COCO, and with custom-trained weights previously obtained on CGMU dataset. The results (see Figure 4.3) showed that with custom-trained weights, more pedestrians were identified. It is possible to notice that the model with pre-trained weights compared to the model with custom-trained fails to identify pedestrians further away from the location of the CCTV camera.

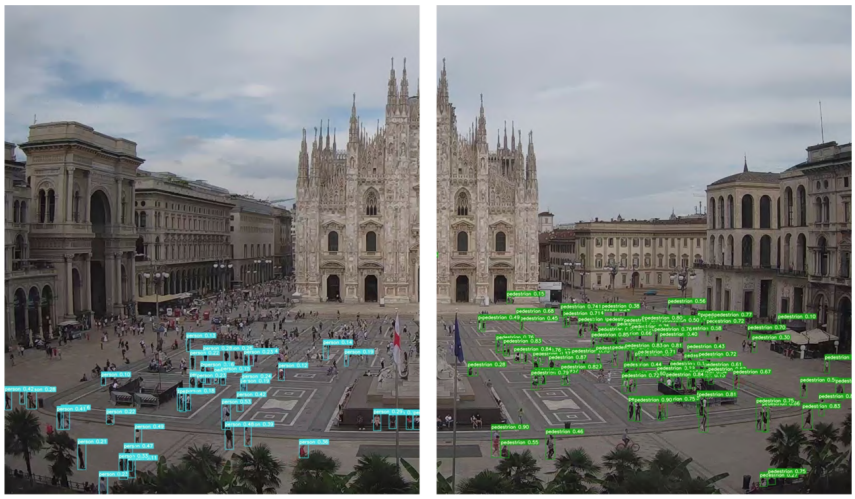


Figure 4.3: Detections with pre-trained weights (left) and with custom-trained weights (right)

This result indicates that the training process of the model on a specific dataset, in this case CGMU, was useful and therefore allows better generalisation on new data.

This is also confirmed by analysing Table 4.2, which shows the average number of pedestrians identified per frame for all five time slots considering pre-trained and custom-trained weights.

Time Slot	**Avg. num. of people/frame with pre-trained weights**	**Avg. num. of people/frame with custom-trained weights**
08:00 - 08:30	66	84
11:00 - 11:30	113	290
12:45 - 13:15	120	369
15:00 - 15:30	116	339
18:00 - 18:30	125	348

Table 4.2: Average number of people per frame in each time slot with trained and no-trained model

It can be seen that with the weights obtained from the training process, the number of pedestrians identified for all time slots is higher. The lowest percentage variation is obtained for the interval 8:00 - 8:30 with a value of 27.273%. In contrast, the highest percentage variation is obtained for the interval 12:45 - 13:15 with a value of 207.500%. These values indicate that the model with pre-trained weights performs well with few pedestrians and therefore few occlusions. With more pedestrians and more occlusions, the model with custom-trained weights performs better.

Focusing on the number of pedestrians obtained with custom-trained weights, it can also be seen that in the first time slot, at 08:00, the number of people in the square is significantly lower than in the other time slots. The time slot with the highest average number of pedestrians per frame is 12:45.

The following phase involved the application of YOLOv7 in combination with a tracker on the Piazza Duomo dataset.

4.2.1 Tracking Evaluation

YOLOv7 was tested with two different object trackers, SORT, and StrongSORT. As with the two models belonging to the YOLO family, the same configuration was used for SORT and StrongSORT in order to identify in a robust way which object tracker was better.

The main parameters considered are: *img-size*, *conf-thres*, and *iou-thres*. For *img-size* parameter different values were tested and 1920 was chosen. After several experiments tuning the *conf-thres* and *iou-thres* the final values for these two parameters are 0.10 and 0.45. The value for *iou-thres* is the default value since it's important to have accurate bounding boxes but in order to detect a significant number of pedestrians the *conf-thres* value is lowered. In this way some false detection could be present in the results.

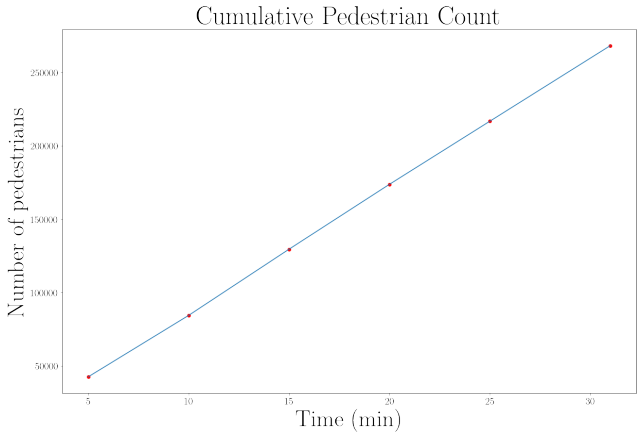
Once detection and tracking have been performed the speed of SORT model compared to strongSORT is considerable. The execution time with the considered hardware resources for a single frame with SORT is about 23 ms while for StrongSORT it is about 4800 ms. This evidence confirms the fact that SORT model is much faster than StrongSORT because it does not have to extract visual features through a CNN as mentioned in (Du et al., 2023). Since the available videos are long, speed is a relevant feature, which is why it was decided to proceed with the SORT model.

Figure 4.4 shows a frame representing the result of the detection and tracking task, using YOLOv7 in combination with the object tracker SORT.

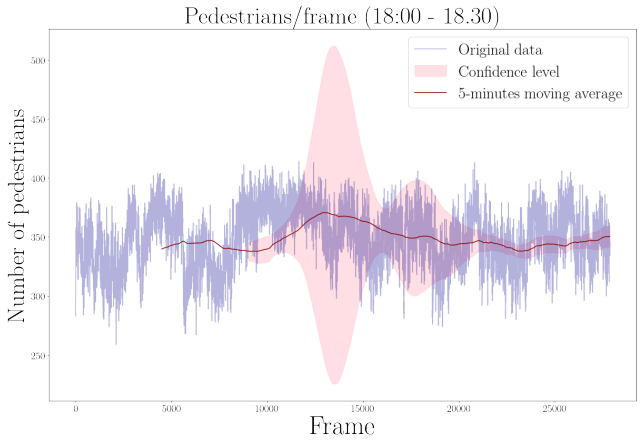
In order to assess the number of pedestrians identified and tracked in the frames constituting the videos, the following graphs were generated: the first representing the cumulative frequency and the second representing the moving average and standard deviation. Both graphs were generated considering a 5-minute interval. The curves for the interval 18:00 - 18:30 are shown below

Figure 4.4: A frame of Piazza Duomo with detection and tracking results for 11:00 - 11:30 time slot

in Figure 4.5.



(a)



(b)

Figure 4.5: (a) Cumulate frequency and (b) Moving average curve for 18:00 - 18:30 time slot

These graphs indicate that the number of pedestrians identified in each frame

has a value with low variability. This observation is also confirmed by analysing the moving average curve. At the same time, the cumulative curve highlights an important aspect related to the id switch issue during detection and tracking. Indeed, as can be seen, the number of unique pedestrians identified every five minutes presents an approximately linear trend. This is an indication that issues such as occlusion are leading to the loss of a large number of existing ids and the creation of new ones.

Focusing on the tracks of the identified pedestrians in the square, it is possible to observe a subset of the extrapolated tracks in Figure 4.6. It can be seen that most of the linear trajectories tend to be located in the external areas of the square. In addition, the presence of the equestrian statue, the street lamps and the subway entrances cause the interruption of some trajectories.

Figure 4.6: A subset of tracks identified in Piazza Duomo for 11:00 - 11:30 time slot

4.3 Geospatial Data Analysis Results

After the detection and tracking steps, the pedestrians were georeferenced as described in 3.7, and the distribution was initially visualised using QGIS as can be seen in Figure 4.7.

This high-level analysis made it possible to understand how the detections are distributed in the square. In addition, this step allowed to verify the correctness of the georeferencing process.

As can be seen from the figure, the detected pedestrians are evenly distributed over the surface of the square, in accordance with the webcam perspective. Also in this case, it can be seen that the pedestrians behind the equestrian statue cannot be identified and tracked. From this plot it can also be seen that the area in front of the entrance to the Milan Cathedral is characterised

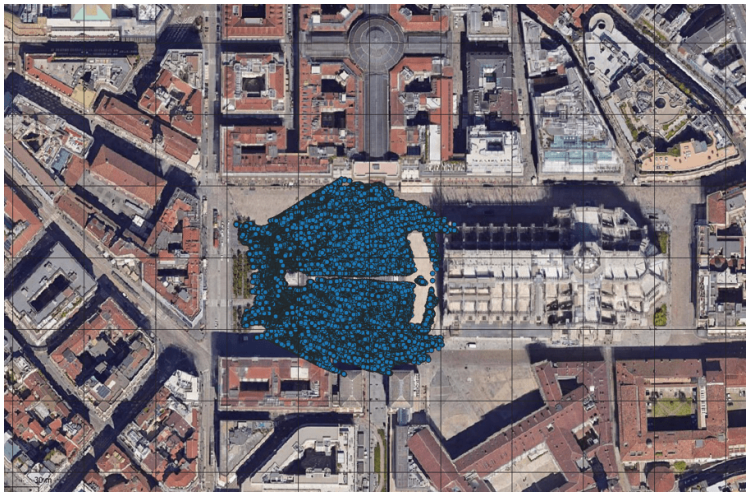
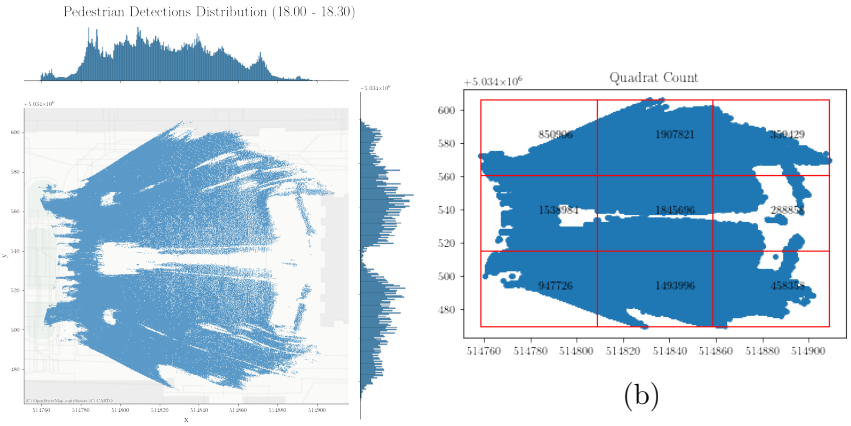


Figure 4.7: QGIS Piazza Duomo results for 11:00 - 11:30 time slot

by the absence of detections due to the fact that this area is cordoned off and cannot be accessed.

4.3.1 Point Pattern Analysis

In order to better understand and analyse the distribution already observed through QGIS, graphs were generated for each time slot using Python scripts. Considering the 18:00 - 18:30 time slot in Figure 4.8, it can be seen that there is a higher concentration of detections in the area of the square close to the CCTV camera due to the fact that pedestrians are more visible and of a larger size. This trend is also noticeable in the other time slots considered.



(a) (b)

Figure 4.8: (a) Pedestrian detections distribution and (b) Quadrat analysis for 18:00 - 18:30 time slot

A KDE heatmap was also generated (see Figure 4.9) for the time slot 18:00 - 18:30, which made it possible to observe the spots in the square with the

highest density. Even in this case, the areas with the highest density are those near the location of the CCTV camera and close to the subway entrances.

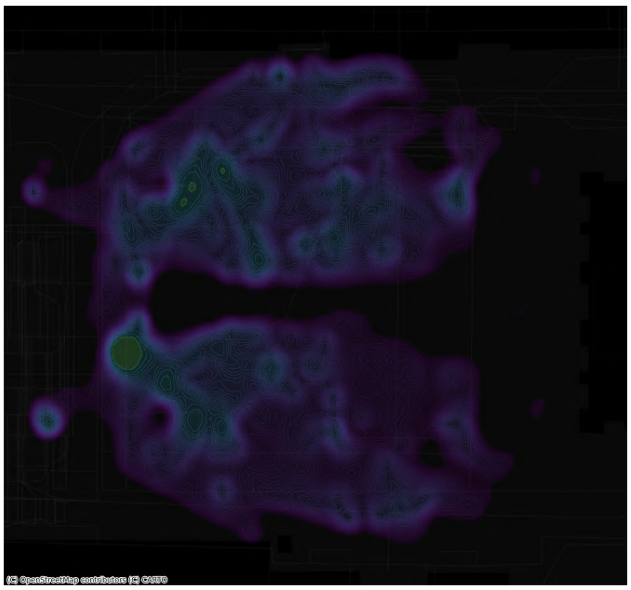


Figure 4.9: KDE heatmap for 18:00 - 18:30 time slot

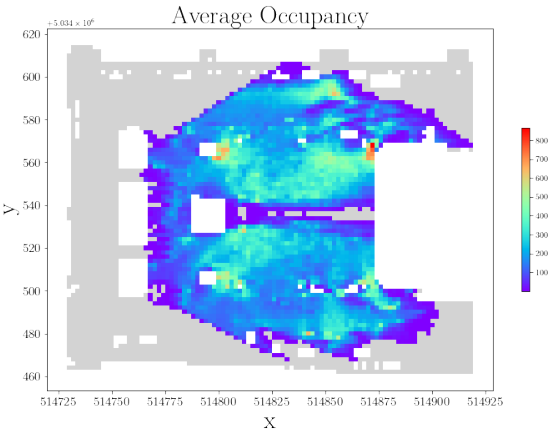
For a more in-depth analysis, the following metrics were calculated for each of the time slots: *Occupancy (pedestrian/cell)*, *Density* (*pedestrian/squared meter*), and *Flow Rate* (*pedestrian/minute/meter*). The results are presented in Table 4.3.

Time Slot	**Occupancy** [$ped/cell$]	**Density** [ped/m^2]	**Flow Rate** [$ped/min/m$]
08:00 - 08:30	57.725 ± 52.355	0.007 ± 0.008	0.948 ± 0.861
11:00 - 11:30	185.599 ± 136.160	0.023 ± 0.021	3.058 ± 2.252
12:45 - 13:15	228.648 ± 154.555	0.029 ± 0.025	3.653 ± 2.472
15:00 - 15:30	223.052 ± 153.374	0.026 ± 0.022	3.669 ± 2.531
18:00 - 18:30	241.459 ± 166.982	0.027 ± 0.025	3.971 ± 2.754
Total	180.495 ± 120.534	0.021 ± 0.017	2.949 ± 1.973

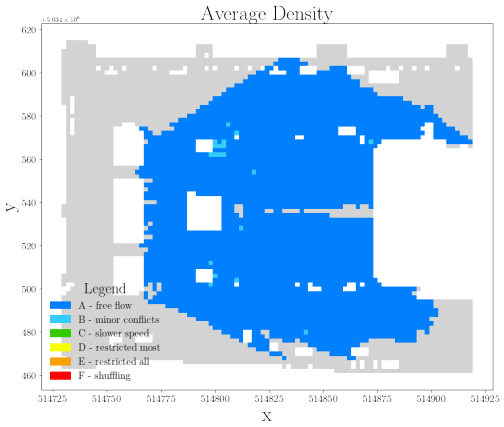
Table 4.3: *Occupancy*, *Density*, *Flow Rate*

Analysing the results, it can be seen that the *Occupancy* value increases during the day, with the highest value (241.459 *pedestrian/cell*) at the time slot 18:00 - 18:30. A similar trend is observed for the other two metrics, *Density* and *Flow Rate*, with lower values in the early hours of the day. These results indicate that the use of the square increases over the hours, with the highest values in the afternoon.

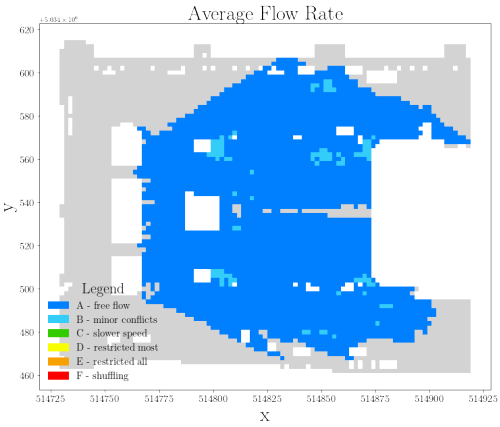
These results can also be visualised through heatmaps, analysing the spatial perspective. Figure 4.10 shows the average scenario, obtained by calculating the mean of the standard metrics previously mentioned considering all time slots.



(a)



(b)



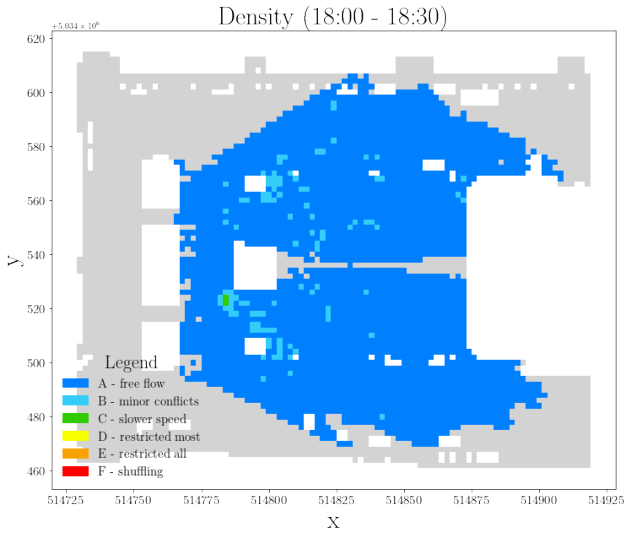
(c)

Figure 4.10: (a) Average *Occupancy (pedestrian/cell)*, (b) Average *Density (pedestrian/squared meter)*, (c) Average *Flow Rate (pedestrian/minute/meter)*

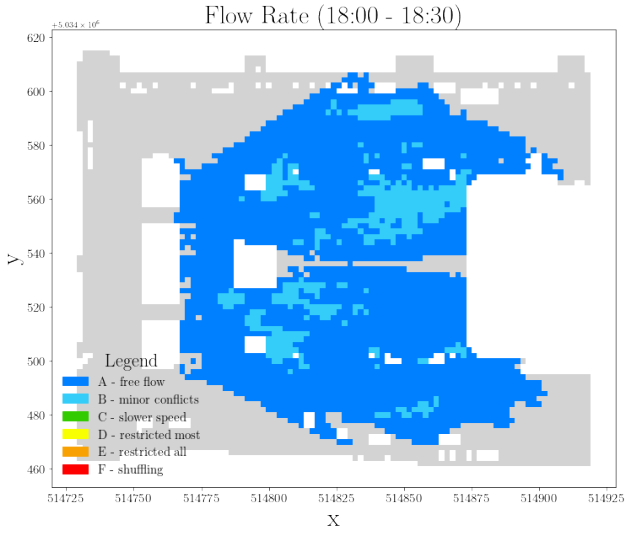
It can be observed that the highest values considering *Occupancy* occur close to the subway accesses in the square. In particular, in these latter areas of the square, *Occupancy* values are higher than 600 *pedestrian/cell*. Regarding *Density* and *Flow Rate* most of the square is characterised by a LOS level A indicating "free flow", considered the average scenario of the five time slots. Some areas of the square are nevertheless characterised by LOS level B representing the presence of "minor conflicts". The LOS level B denotes irregular flow under low- to medium density conditions.

The highest values that characterise the last time slot (as seen in Table 4.3), 18:00 - 18:30, can be visualised in Figure 4.11. There is a greater area of the square characterised by LOS level B compared to the average scenario. In this case there are also values of LOS up to C level, representing areas where pedestrian mobility is rather congested and as a result there is a "slower speed".

The heatmaps related to the metrics discussed for the specific time slots can be consulted in Appendix A (see Figures A.1, A.2, A.3, A.4, A.5). In these



(a)



(b)

Figure 4.11: (a) Density and (b) Flow Rate for 18:00 - 18:30 time slot

graphs, it can be seen how the use of the square increases over the course of the day. In particular, looking at the *Flow Rate*, there is a clear increase from a value of 0.948 *pedestrian/minute/meter* at 08:00 to a value of 3.971 *pedestrian/minute/meter* at 18:00.

4.3.2 Trajectory Data Mining

Once the analysis of the distribution of individual detections in the square was completed, the trajectories identified were analysed.

The first trajectory filtering operation consisted of considering only trajectories whose duration was higher than 20 seconds. Considering the five time slots, the average number of detections and trajectories eliminated were 78.247%

and 97.609% of the total respectively. These percentages are very high since there is a lot of noise in the observations due to occlusion issues.

The second main filtering step consisted of discarding trajectories containing points with speeds above 2 m/s, which were considered outliers. From the previously processed trajectories a substantial deletion was again carried out in order to have clean and meaningful data. In particular, in this step an average of 76.580% of the detections and 75.999% of the trajectories were eliminated, considering the five time slots. Using a threshold of 1.5 m/s, the elimination rates of detections and trajectories would have been even higher (on average 87.550% and 87.415%), leading to discarding significant trajectories.

The Figure 4.12 shows graphically for all the time slots the resulting trajectories and their respective points.

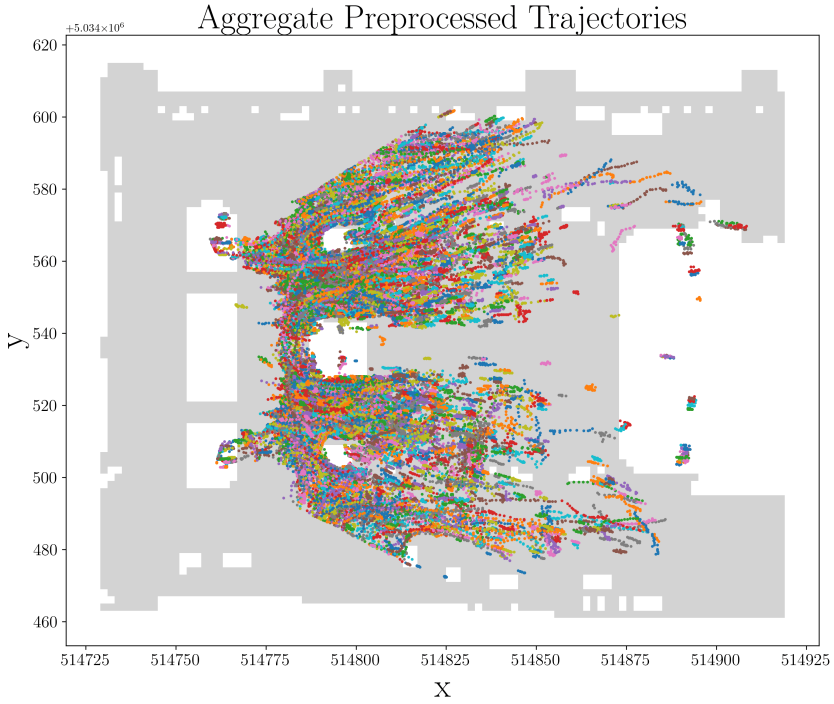


Figure 4.12: Overall preprocessed trajectories

As seen above, detections are concentrated in the first area of the square, which is closer to the position of the installation of CCTV camera. The Figure A.6 in Appendix A shows the resulting preprocessed trajectories for each time slot.

Analysing the data obtained in a more quantitative manner, for each time slot average values of time distance, duration, speed, and direction, are presented in Table 4.4.

At the end of the data cleaning operations for the five time slots, 4,326 trajectories were obtained: 255 for the first timeslot, 922 for the second, 1,274 for the third, 970 for the fourth, and 905 for the last time slot. The first time

Time Slot	Count	Avg. Distance $[m]$	Avg. Duration $[s]$	Avg. Speed $[m/s]$	Avg. Direction $[°]$
08:00 - 08:30	255	18.087 ± 10.472	31.751 ± 15.226	0.610 ± 0.377	120.422 ± 57.002
11:00 - 11:30	922	13.236 ± 8.332	32.150 ± 18.740	0.436 ± 0.274	113.616 ± 46.916
12:45 - 13:15	1,274	13.589 ± 8.259	31.147 ± 14.354	0.450 ± 0.271	116.79 ± 47.385
15:00 - 15:30	970	13.653 ± 8.852	30.046 ± 13.367	0.468 ± 0.296	121.909 ± 53.729
18:00 - 18:30	905	12.523 ± 8.525	31.235 ± 16.279	0.418 ± 0.281	116.899 ± 47.671

Table 4.4: Trajectories summary statistics

slot, 08:00 - 08:30, is characterised by the lowest number of trajectories but at the same time by the highest speed value. It can be assumed that there are mostly commuters at this time of day, i.e. people on their way to work. Over the hours assuming that the number of tourists increases as found in the following clustering results, the number of identified trajectories increases and the average speed decreases. This indicates greater pedestrian congestion in the square. In the last time slot, 18:00 - 18:30, the lowest values for speed and distance travelled are reported.

Considering the preprocessed trajectories, clustering was performed to identify the trajectories of commuters and tourists. Once the two clusters were obtained, it was necessary to assign the respective label, commuter and tourist. The following choice in absence of ground truth was made on the characteristics of the two clusters by analysing the average values of distance, duration, speed, and direction presented in Table 4.5, and secondly observing the results also graphically, taking into consideration the spatial component with the generated graphs.

Time Slot	Avg. Distance $[m]$		Avg. Duration $[s]$	
	Commuters	*Tourists*	*Commuters*	*Tourists*
08:00 - 08:30	26.956 ± 7.269	10.802 ± 6.209	28.012 ± 9.149	34.822 ± 18.274
11:00 - 11:30	23.007 ± 7.233	9.125 ± 4.424	31.869 ± 16.742	32.268 ± 19.530
12:45 - 13:15	22.902 ± 7.210	9.778 ± 4.984	28.895 ± 10.607	32.069 ± 15.541
15:00 - 15:30	24.488 ± 7.990	9.727 ± 5.068	27.430 ± 8.129	30.994 ± 14.705
18:00 - 18:30	23.630 ± 7.689	8.717 ± 4.534	29.745 ± 14.101	31.746 ± 16.941

Time Slot	Avg. Speed $[m/s]$		Avg. Direction $[°]$	
	Commuters	*Tourists*	*Commuters*	*Tourists*
08:00 - 08:30	0.960 ± 0.240	0.322 ± 0.166	156.568 ± 61.191	90.731 ± 29.962
11:00 - 11:30	0.763 ± 0.240	0.298 ± 0.136	149.558 ± 57.163	98.497 ± 31.363
12:45 - 13:15	0.792 ± 0.204	0.310 ± 0.138	155.572 ± 59.831	100.910 ± 28.875
15:00 - 15:30	0.872 ± 0.203	0.322 ± 0.155	177.725 ± 61.486	101.684 ± 32.058
18:00 - 18:30	0.814 ± 0.220	0.282 ± 0.131	162.510 ± 59.800	101.267 ± 29.524

Table 4.5: Clustering results

Hypothesising and referring to previous works (Gorrini et al., 2016), it can be stated that the trajectories of commuters are characterised by higher linearity and higher speed, unlike those of tourists which are characterised by greater uncertainty and lower speed. Based on these assumptions, the labels were assigned to the respective clusters. The average percentage of tourists walking

the square (68.825% ± 50.744) is higher than the average percentage of commuters (31.175% ± 50.744) considering the five time intervals. The variance values are very high due to the first time slot which has a significantly lower number of pedestrians in the square. Focusing on this first interval, 08:00 - 08:30, the number of pedestrians identified as commuters is 115 while the number of pedestrians identified as tourists is 140. In comparison with the other time slots in which the percentage of tourists is always higher than 70%, in this case the difference between the two categories of pedestrians is minimal. This can be explained by the fact that being early in the morning, the number of tourists in the square is low and will increase over the day.

Examining the results in the Table 4.5, it can be seen that the distance, speed, and direction values are higher for commuters than for the trajectories associated to the tourists' cluster. The latter is in fact characterised by more irregular trajectories and a lower speed, which leads to covering shorter distances in more time. An example of the two clusters on trajectories of commuter and tourist trajectories related to 18:00 time slot is shown in Figure 4.13. As can be seen from the image, trajectories identified as belonging to the commuters cluster appear to be longer and more linear. Trajectories associated to tourists, on the other hand, appear shorter and more randomly distributed.

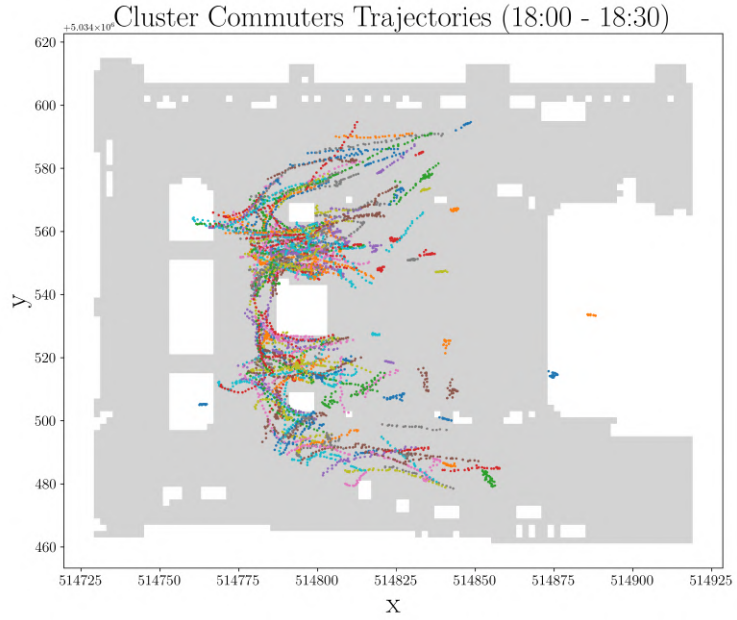
In order to better understand and validate the differences between the pedestrians identified as commuters and those identified as tourists, two-tailed t-tests with independent samples were performed. The t-tests confirmed, as can be seen in Table 4.6, that the average values of distance, duration, speed, and direction are significantly different for the two types of pedestrians except for duration values in the 11:00 and 18:00 time slots.

Time Slot	**Avg. Distance** *Commuters vs Tourists t-test*	**Avg. Duration** *Commuters vs Tourists t-test*	**Avg. Speed** *Commuters vs Tourists t-test*	**Avg. Direction** *Commuters vs Tourists t-test*
08:00 - 08:30	p value<.001	p value<.001	p value<.001	p value<.001
11:00 - 11:30	p value<.001	-	p value<.001	p value<.001
12:45 - 13:15	p value<.001	p value<.001	p value<.001	p value<.001
15:00 - 15:30	p value<.001	p value<.001	p value<.001	p value<.001
18:00 - 18:30	p value<.001	-	p value<.001	p value<.001

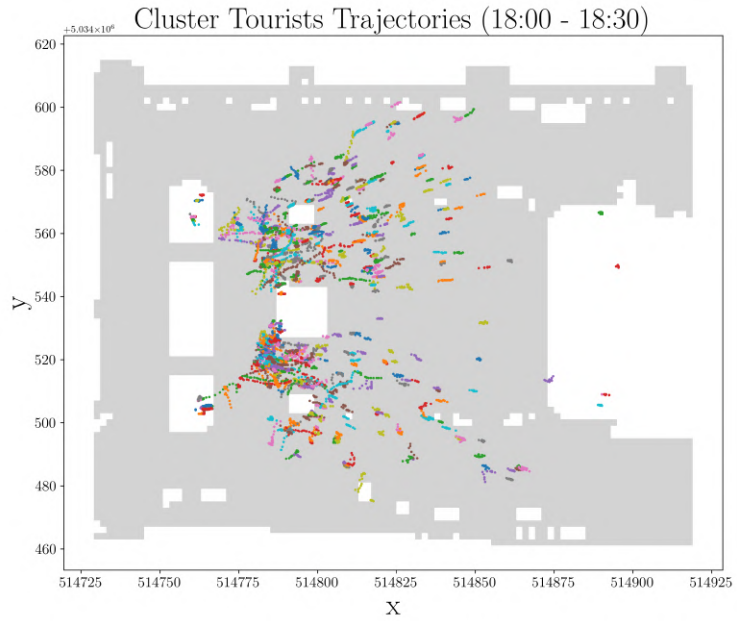
Table 4.6: Commuters vs. Tourists: independent-samples two-tails t-test

Considering 11:00 - 11:30 time slot, 273 commuters ($M = 31.869$, $SD = 16.742$) compared to 649 tourists ($M = 32.268$, $SD = 19.53$) demonstrated a non-significant difference considering duration, $t(920) = -0.295$, $p = .768$. Similarly, considering the duration as well, in this case for the time slot 18:00 - 18:30, 231 commuters ($M = 29.745$, $SD = 14.101$) compared to 674 tourists ($M = 31.746$, $SD = 16.941$) do not indicate a significant difference, $t(903) = -1.613$, $p = .107$.

The further analysis allowed to investigate the presence or absence of groups, i.e. pedestrians whose trajectories have close (no more than 4 m) start and end points and the remaining points close (no more than 1.5 m) for the duration of their existence at the same time instants. In Figure 4.14, for the interval 18:00



(a)



(b)

Figure 4.13: (a) Commuters and (b) Tourists clusters for 18:00 - 18:30 time slot

- 18:30, a subset of pedestrian trajectories identified as being part of groups is presented.

Groups of different sizes were obtained but in order to conduct analyses with higher accuracy it was decided to consider only two- and three-members groups. Analysing the results (see Figure 4.15) considering all time slots, two-members groups represent the majority. The fact that the number of groups of more

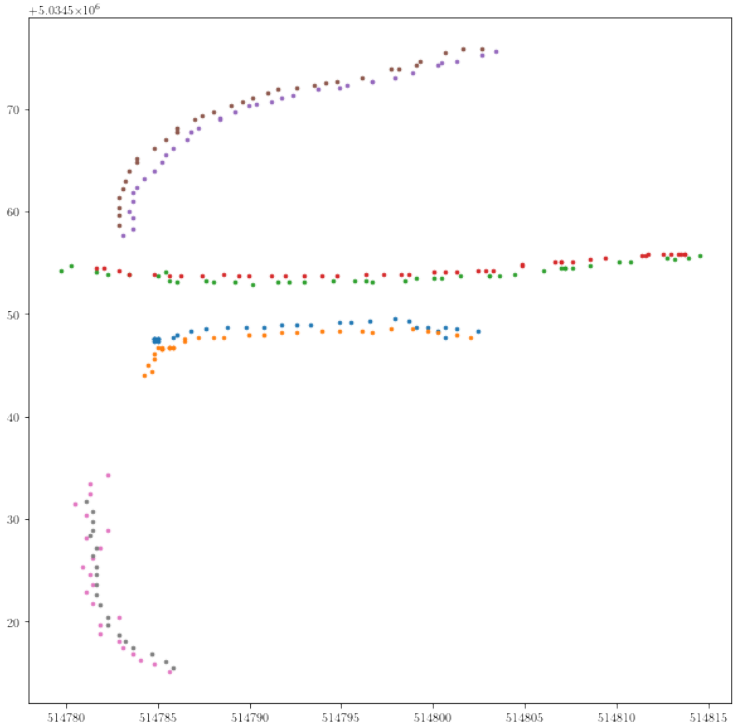


Figure 4.14: A subset of two-members groups detected in time slot 18:00 - 18:30

than two people is significantly lower is likely caused by the switch ID issue which makes it more difficult to find groups characterised by a higher number of people. Focusing on the single time slot, the 15:00 - 15:30 interval is the one with the highest number of identified groups. The same time slot has the highest number of identified single pedestrians.

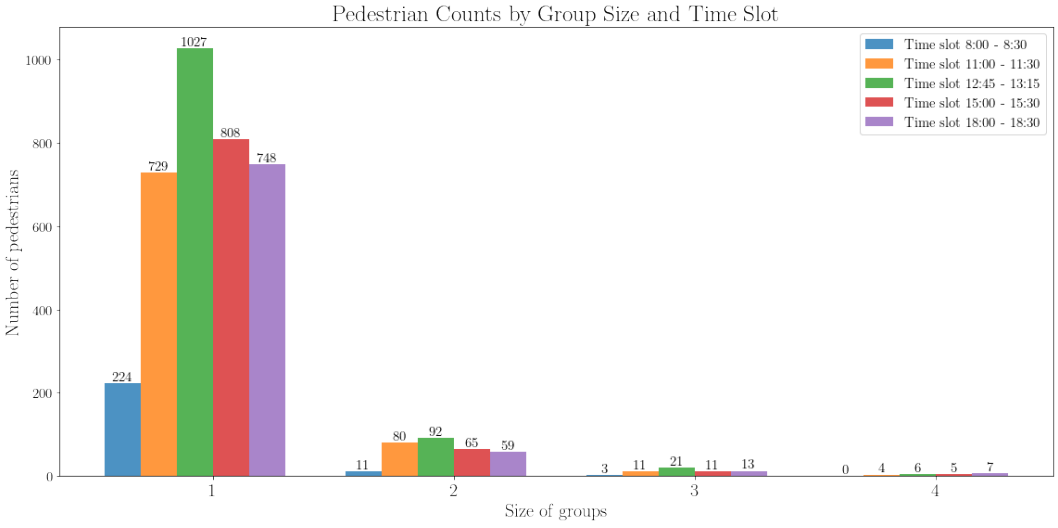


Figure 4.15: Frequency of identified groups according to their types (groups of size 1 means individuals).

The results related to distance, duration, speed and direction features considering single pedestrians and groups of two or three people are shown in Table 4.7.

Time Slot	**Avg. Distance** $[m]$		**Avg. Duration** $[s]$	
	Single pedestrians	*Groups*	*Single pedestrians*	*Groups*
08:00 - 08:30	19.323 ± 10.478	9.156 ± 4.320	31.645 ± 15.501	32.516 ± 13.269
11:00 - 11:30	14.060 ± 8.664	10.123 ± 6.006	30.829 ± 15.044	37.142 ± 28.192
12:45 - 13:15	14.229 ± 8.408	10.928 ± 7.022	30.034 ± 13.468	35.777 ± 16.822
15:00 - 15:30	14.344 ± 9.137	10.208 ± 6.234	29.559 ± 13.348	32.479 ± 13.237
18:00 - 18:30	12.981 ± 8.698	10.343 ± 7.287	30.589 ± 14.630	34.316 ± 22.345
Time Slot	**Avg. Speed** $[m/s]$		**Avg. Direction** $[^\circ]$	
	Single pedestrians	*Groups*	*Single pedestrians*	*Groups*
08:00 - 08:30	0.655 ± 0.378	0.280 ± 0.109	124.298 ± 58.713	92.416 ± 31.037
11:00 - 11:30	0.471 ± 0.285	0.301 ± 0.171	116.754 ± 49.238	101.763 ± 34.457
12:45 - 13:15	0.483 ± 0.276	0.310 ± 0.195	120.643 ± 48.918	100.746 ± 36.307
15:00 - 15:30	0.496 ± 0.301	0.329 ± 0.226	125.572 ± 55.086	103.640 ± 41.973
18:00 - 18:30	0.440 ± 0.290	0.309 ± 0.203	119.304 ± 49.783	105.444 ± 33.791

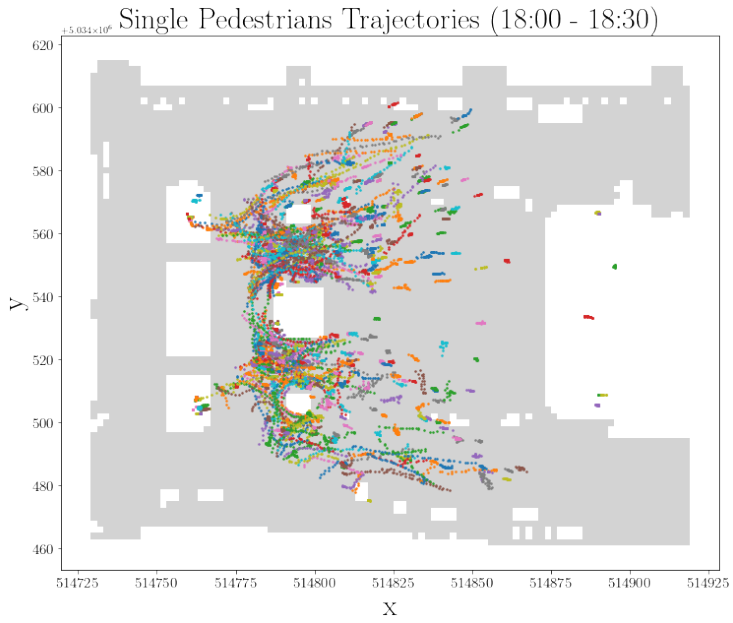
Table 4.7: Groups detection results

The number of pedestrians identified as being part of a group are fewer than the number of individual trajectories for all the five time slots considered. The average percentage of pedestrians belonging to groups (consisting of two or three people) is 17.305% ± 2.979 of all pedestrians in the square while the value for single pedestrians is 82.695% ± 2.979 considering the five time intervals. The average distance traveled by both single pedestrians and groups is relatively consistent across all time slots. Considering the average duration of trajectories, it can be seen that groups tend to spend slightly more time traveling than single pedestrians. Regarding speed, the data indicates that single pedestrians cross the square at a higher speed than pedestrians identified as part of groups, with the highest value of 0.655 m/s in 08:00 time slot.

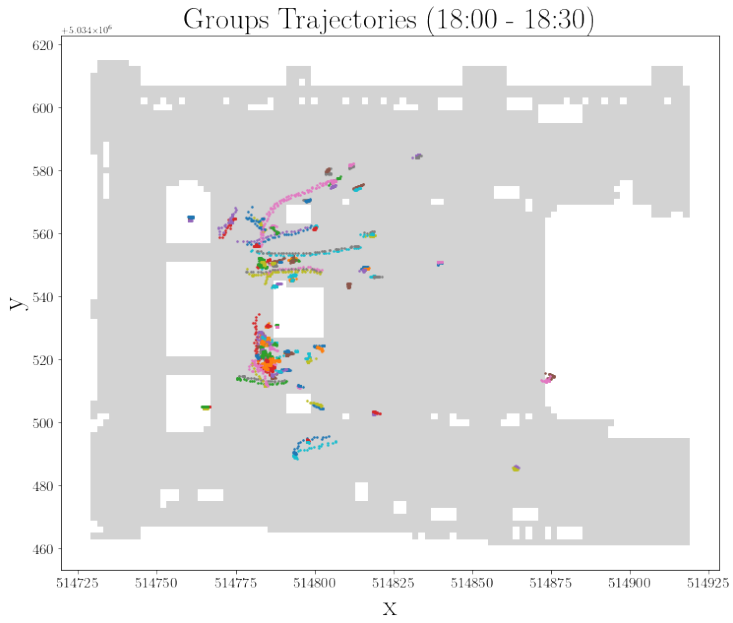
An example of trajectories categorised into single pedestrians and groups for the 18:00 - 18:30 time slot is shown in Figure 4.16. As can be seen from the Figure, the identified groups tend to be located in the first section of the square, near the statue and the subway accesses. It can also be seen in the Figure that there are fewer groups than single pedestrians.

A similar procedure was then followed as previously by generating t-tests to confirm with more confidence the differences identified between single pedestrians and groups. Examining Table 4.8, it can be seen that the average values of the considered metrics (distance, duration, speed, direction) are significantly different for single pedestrians and groups considering the five time intervals except for the duration value in the first time slot. In the interval 08:00 - 08:30, considering duration, 224 single pedestrians ($M = 31.645$, $SD = 15.501$) compared to 31 pedestrians belonging to groups ($M = 32.516$, $SD = 13.269$) demonstrated a non-significant difference, $t(255) = -0.298$, $p = .766$.

The results obtained for the clustering and group detection process are sum-



(a)



(b)

Figure 4.16: (a) Single pedestrians and (b) Groups for 18:00 - 18:30 time slot

Time Slot	**Avg. Distance** *Single pedestrians vs Groups t-test*	**Avg. Duration** *Single pedestrians vs Groups t-test*	**Avg. Speed** *Single pedestrians vs Groups t-test*	**Avg. Direction** *Single pedestrians vs Groups t-test*
08:00 - 08:30	p value<.001	-	p value<.001	p value<.005
11:00 - 11:30	p value<.001	p value<.001	p value<.001	p value<.001
12:45 - 13:15	p value<.001	p value<.001	p value<.001	p value<.001
15:00 - 15:30	p value<.001	p value<.005	p value<.001	p value<.001
18:00 - 18:30	p value<.001	p value<.005	p value<.001	p value<.001

Table 4.8: Single pedestrians vs. Groups: independent-samples two-tails t-test

marised in Table 4.9 below. The "Detected groups" value refers to the number of groups consisting of two or three people.

Time slot	Total points	Total trajectories	Detected commuters	Detected tourists	Detected single pedestrians	Detected groups
08:00 - 08:30	7,966	255	115	140	224	14
11:00 - 11:30	29,151	922	273	649	729	91
12:45 - 13:15	39,058	1,274	370	904	1,027	113
15:00 - 15:30	28,719	970	258	712	808	76
18:00 - 18:30	27,826	905	231	674	748	72

Table 4.9: Clustering and Groups detection results

As can be seen, the highest number of detections and trajectories identified is in the 15:00 - 15:30 time slot. This time slot also has the highest number of commuters and tourists. The same occurs also for single pedestrians and groups. On the other hand, the time slot that has the lowest values in terms of points and trajectories is the first time slot in the morning, 08:00 - 08:30.

4.4 Discussion

In this section a detailed discussion of the results is provided, contextualising them with regard to the research questions and the relevant literature.

The training of the object detection model was useful in order to obtain more accurate results. Indeed, by comparing the results of the untrained model with those of the trained model, the number of pedestrians identified increased significantly as can also be seen in Figure 4.3. Nevertheless, both the detection and tracking phases revealed problems such as a large number of pedestrian switches due to occlusion and, in particular, to the small dimension of the pedestrians considered. Reviewing the literature, detection and tracking small objects especially in dense crowds is still a complex and open issue. In such situations, computer vision techniques are likely to fail due to the fact that pedestrians are represented by a limited number of pixels and many occlusion events occur. Regarding the tracking step, results could have been improved by considering a training phase for the tracker model as well. Nevertheless, it was noticed that there was a considerable lack of suitable datasets for this purpose.

The detailed analysis of pedestrian detections enabled a mobility study to characterise the urban space considered, in this case, a public square. The features and the distribution patterns were analysed through maps and the use of metrics that are typically used in transport engineering, particularly for pedestrian flows. Considering the results obtained (see Figure 4.10) it could be seen that the metrics are able to characterise the urban space examined to a limited extent. In order to acquire a deeper understanding of the analysed area and to have more insights metrics specifically designed for such scenarios would be required. This need emerges since the metrics considered are more appropriate for more restricted spaces characterised by single-directionality

and a lower extension. In this particular case, Piazza Duomo represents a large and complex space consisting of multidirectional agents.

Lastly, focusing on individual pedestrian trajectories, an attempt was made to identify different pedestrian utilisation profiles and the presence of pedestrian groups. In order to be able to perform such analyses, it was first necessary to perform preprocessing steps which resulted in the removal of several trajectories. This reduction of data on the one hand may have caused the deletion of potentially meaningful trajectories, but on the other hand was necessary for computational purposes and in order to be able to interpret the results more clearly. The distinction in commuters and tourists was based on the analysis of the main characteristics of the trajectories. By carrying out statistical analyses and with the support of maps, it was possible to observe the presence of different pedestrians that can be associated with commuters and tourists as mentioned before. Significant differences were discovered between the two pedestrian profiles in terms of distance, duration, speed and direction, considering all time slots. No significant differences were only found between commuters and tourists for the 11:00 and 18:00 time slots considering the duration metric. Furthermore, analysing the number of pedestrians identified as commuters and tourists it was noted that there is no clear difference for the early morning time slot. This is due to the fact that tourists tend to gather in the later time slots and not in the early morning as it would be expected. This can also be seen by observing the preprocessed trajectories for the interval 08:00 - 08:30 (see Figure A.6a) characterised by a longer distance travelled and higher speed, which are more likely to be attributed to commuters. The difference in the number of commuters and tourists becomes marked in the other time slots of the day.

A further analysis was carried out to try to identify among the trajectories those relating to pedestrians associated with groups. Analysing the identified groups, a particular emphasis was on groups consisting of two or three people. It was found that the groups consisting of two people were the most frequent. As previously, statistical analyses and maps revealed significant differences between single pedestrians and pedestrians belonging to groups. More specifically, it was found that the trajectories of pedestrians belonging to groups were characterised by a shorter distance travelled, a higher duration in seconds, a lower speed, and lower direction values. Also in this case, for the time slot 08:00 - 08:30 and for the duration metric only, no significant differences between the trajectories of grouped and single pedestrians were found.

Chapter 5

Conclusions and Future Work

The aim of the thesis was to prove the potential of Urban Informatics in a real-life scenario. It involved the application of innovative computer vision techniques based on the concept of deep learning in the context of urban planning. The objective was to use CCTV camera videos of Piazza Duomo, one of Italy's most important squares to detect and track pedestrians in order to extrapolate meaningful mobility patterns. The intent was to use urban planning metrics to characterise the urban space examined and identify different categories of pedestrians, including the presence or absence of groups, by using trajectories data collected through tracking video analytics techniques.

The results obtained clearly revealed the high potential of computer vision in conducting urban mobility analysis. The major contribution in applying such techniques consists in accelerating these operations that otherwise would require considerable time and resources of human analysts. The use of computer vision techniques could therefore save time in the initial stages and can be seen as a support tool for the analyst who could focus more on the analysis steps. If on one hand these innovative techniques can guarantee more accuracy and robustness in the urban planning analysis on the other relying on artificial intelligence techniques requires always a cautiously and critically supervision by urban planners. There is certainly a need to conduct further research in this field to be able to achieve satisfying results, even in complex situations. Indeed, the algorithms used for the detection and tracking phases perform very well in no-dense crowd situations and with large pedestrians. Indeed, in the examined scenario where the pedestrians to be identified and tracked are very small and there are crowded situations, the performance of these algorithms is not optimal. Furthermore, it emerged that there is a need for a larger number of annotated datasets reflecting also complex scenarios. The results obtained are relatively good considering the complexity of the dataset that was used, i.e. a dataset without annotations and characterised by very small and low-resolution pedestrians. The urban planning metrics used have enabled a good characterisation of urban space, nevertheless evidencing some limitations. These metrics clearly perform better in more restricted areas. The

pedestrian trajectory analysis allowed a reasonably accurate categorisation of pedestrians. Based on the overall features of the trajectories, it was indeed possible to identify potential commuters and potential tourists trough the use of clustering algorithms. The results obtained were also validated by means of statistical techniques such as t-tests. Similarly, the identification of groups of people was carried out with meaningful results. Overall, the obtained results allowed from raw trajectory data collected by detection and tracking algorithms to characterise pedestrian profiles and identify the presence of groups of pedestrians within the square.

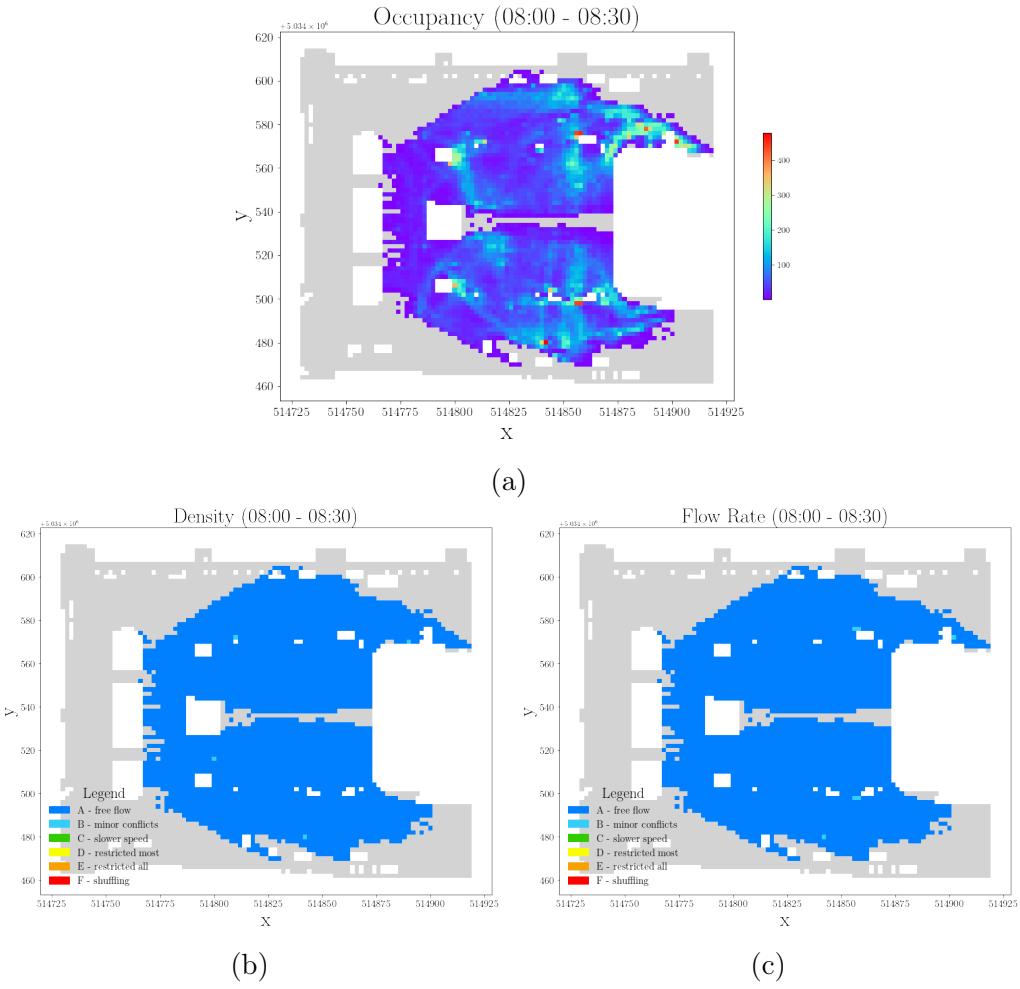
Lastly, the results of this research could be relevant for the purposes of evidence-based/ parametric approach for regeneration projects of urban public spaces. Furthermore, in the calibration/ validation process of computer-based pedestrian modelling and simulation systems, the proposed methodology could be useful.

Future developments could involve investing more time and resources in the training phase of the detection and tracking algorithms in order to obtain more accurate results. Regarding the lack of annotated datasets for tracking small pedestrians in crowded scenes it could be considered to manually annotate one of the available datasets and see if this can provide improvements. Alternatively, pedestrian micro-simulations that recreate complex and realistic scenarios could be used. In relation to the tracking phase, the approach used in this work does not exploit the full potential of deep learning due to limited computational capabilities. As a future development, a more efficient tracker model could be considered. Furthermore, identified bounding boxes could be examined more in detail, starting from their size inferences about age of pedestrians could be drawn. Future work will be focused also on a deeper research and development of new metrics appropriate for such scenarios. To validate the results obtained in a more robust way, it might be useful to carry out inferences on datasets representing different scenarios.

Appendix A

A.1 Occupancy, Density, and Flow Rate Heatmaps

The following maps represent *Occupancy*, *Density* and *Flow Rate* related to LOS values considering all time intervals. It can be seen that the use of the square increases over the hours.

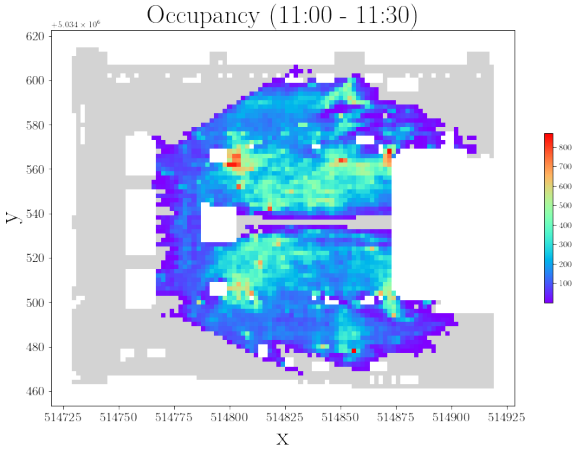


(a)

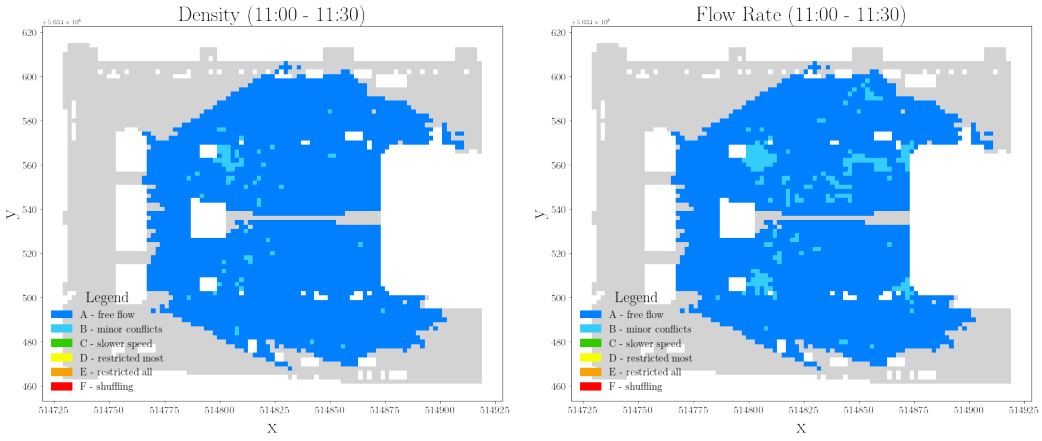
(b)

(c)

Figure A.1: (a) *Occupancy*, (b) *Density*, (c) *Flow Rate* for 08:00 - 8:30 time slot

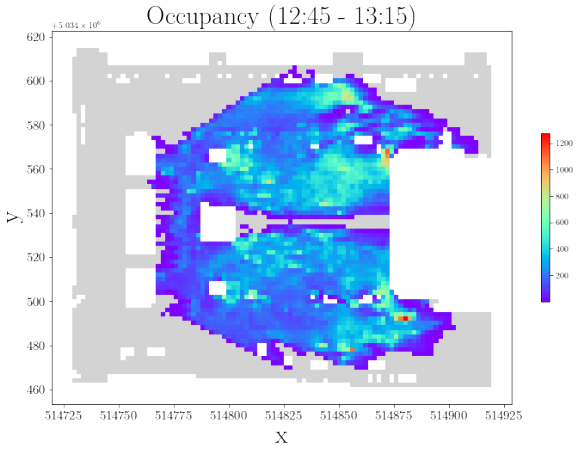


(a)

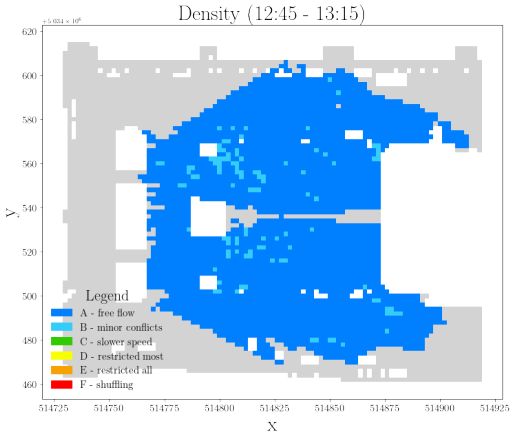


(b) (c)

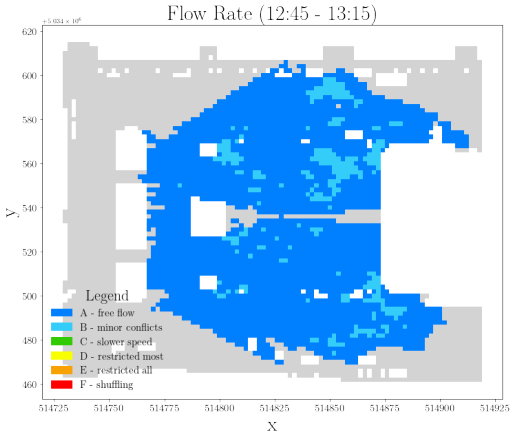
Figure A.2: (a) *Occupancy*, (b) *Density*, (c) *Flow Rate* for 11:00 - 11:30 time slot



(a)

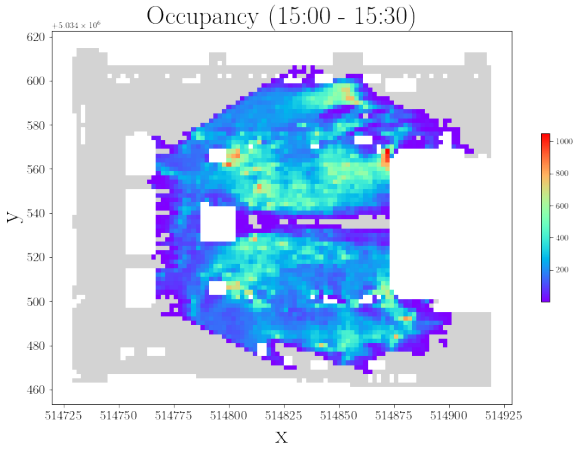


(b)

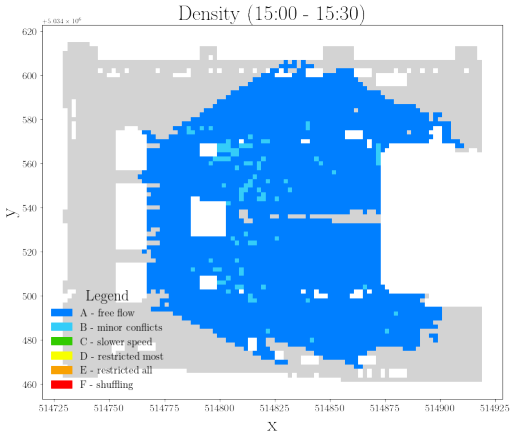


(c)

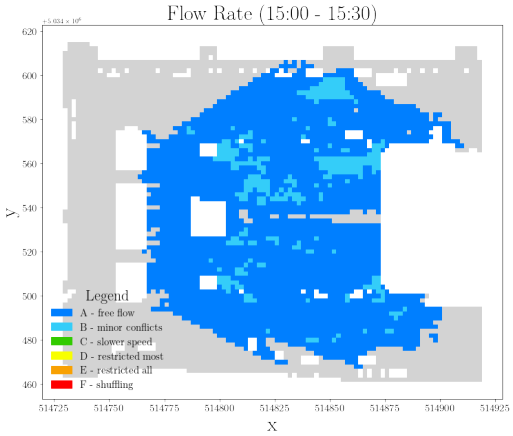
Figure A.3: (a) *Occupancy*, (b) *Density*, (c) *Flow Rate* for 12:00 - 12:30 time slot



(a)

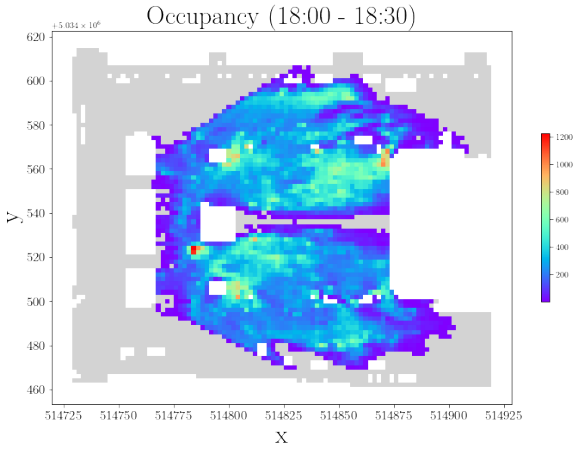


(b)

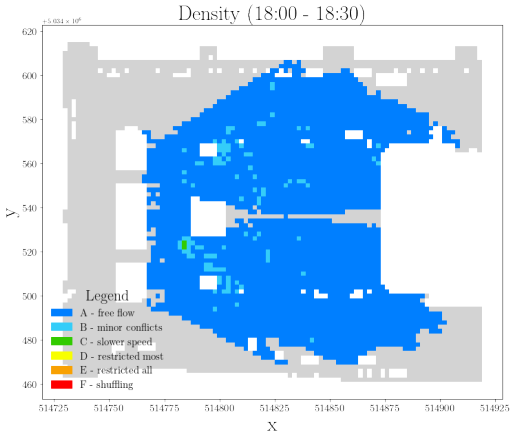


(c)

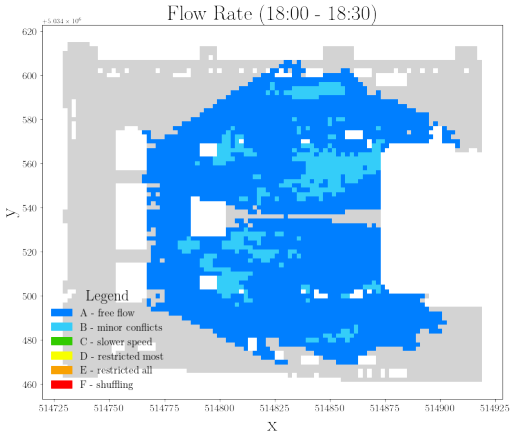
Figure A.4: (a) *Occupancy*, (b) *Density*, (c) *Flow Rate* for 15:00 - 15:30 time slot



(a)



(b)

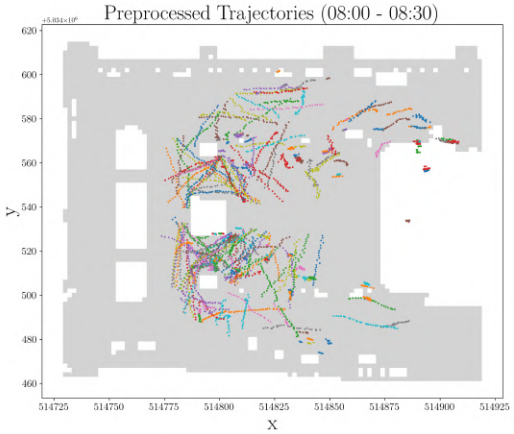


(c)

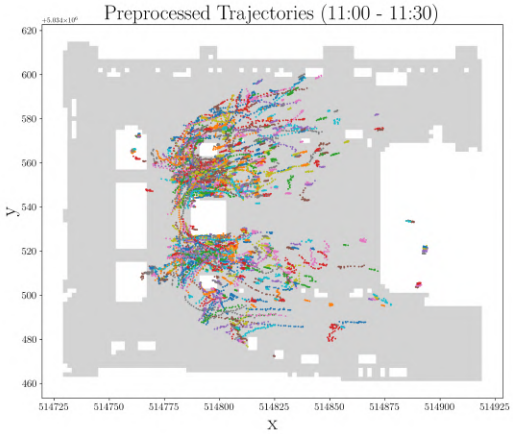
Figure A.5: (a) *Occupancy*, (b) *Density*, (c) *Flow Rate* for 18:00 - 18:30 time slot

A.2 Preprocessed Trajectories

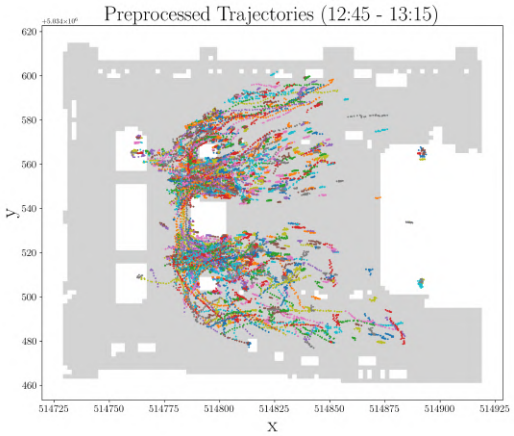
The following maps represent the overall preprocessed trajectories considering all time intervals. The number of preprocessed trajectories increases over the hours



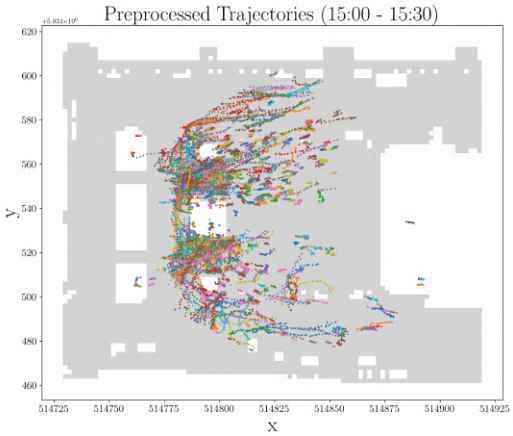
(a) 08:00 - 08:30 time slot



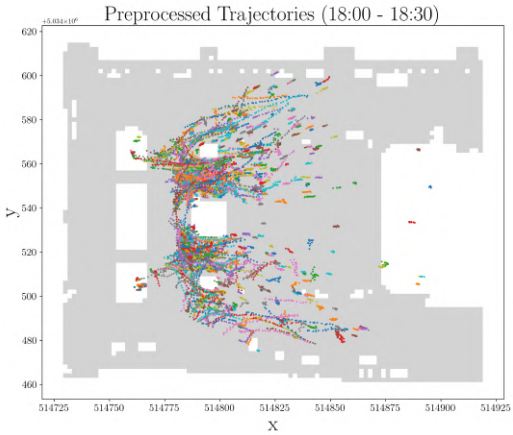
(b) 11:00 - 11:30 time slot



(c) 12:00 - 12:30 time slot



(d) 15:00 - 15:30 time slot



(e) 18:00 - 18:30 time slot

Figure A.6: Preprocessed trajectories

Bibliography

Akyon, F. C., Altinuc, S. O., and Temizel, A. (2022). Slicing aided hyper inference and fine-tuning for small object detection. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 966–970. IEEE.

Almohaimeed, N. and Prince, M. (2019). A comparative study of different oject tracking methods in a video. *International Journal of Computer Applications*, 975:8887.

Andrienko, G., Andrienko, N., Bak, P., Keim, D., and Wrobel, S. (2013). *Visual analytics of movement*. Springer Science & Business Media.

Aziz, L., Salam, M. S. B. H., Sheikh, U. U., and Ayub, S. (2020). Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review. *IEEE Access*, 8:170461–170495.

Batty, M. (2010). The pulse of the city.

Bergmann, P., Meinhardt, T., and Leal-Taixe, L. (2019). Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 941–951.

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE.

Bian, J., Tian, D., Tang, Y., and Tao, D. (2018). A survey on trajectory clustering analysis. *arXiv preprint arXiv:1802.06971*.

Bierlaire, M. and Robin, T. (2009). Pedestrians choices. In *Pedestrian behavior*. Emerald Group Publishing Limited.

Boeing, G., Batty, M., Jiang, S., and Schweitzer, L. (2022). Urban analytics: History, trajectory and critique. In *Handbook of Spatial Analysis in the Social Sciences*, pages 503–516. Edward Elgar Publishing.

Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface. .

Brunetti, A., Buongiorno, D., Trotta, G. F., and Bevilacqua, V. (2018). Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, 300:17–33.

Cavallaro, C. and Vizzari, G. (2022). A novel spatial–temporal analysis approach to pedestrian groups detection. *Procedia Computer Science*, 207:2364–2373.

Ceccarelli, G., Messa, F., Gorrini, A., Presicce, D., and Choubassi, R. (2023a). Deep learning video analytics for the assessment of street experiments: The case of bologna. *Journal of Urban Mobility*. Submitted.

Ceccarelli, G., Presicce, D., and Deponte, D. (2023b). *Looking with Machine Eyes: City Monitoring for Urban Resilience*. Springer. In press.

Cheng, G., Yuan, X., Yao, X., Yan, K., Zeng, Q., and Han, J. (2022). Towards large-scale small object detection: Survey and benchmarks. *arXiv preprint arXiv:2207.14096*.

Ciaparrone, G., Sánchez, F. L., Tabik, S., Troiano, L., Tagliaferri, R., and Herrera, F. (2020). Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88.

City of Montreal (2020). Images annotées - caméras de circulation. `https://donnees.montreal.ca/ville-de-montreal/images-annotees-cameras-circulation`.

Crociani, L., Gorrini, A., Feliciani, C., Vizzari, G., Nishinari, K., and Bandini, S. (2019). Micro and macro pedestrian dynamics in counterflow: The impact of social group. In *Traffic and Granular Flow'17 12*, pages 151–158. Springer.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee.

Derrick, I. A. (1999). Pedestrian detection by computer vision. Degree of doctor of philosophy, Edinburgh Napier University.

Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., and Meng, H. (2023). Strongsort: Make deepsort great again. *IEEE Transactions on Multimedia*.

Fan, S., Zhu, F., Chen, S., Zhang, H., Tian, B., Lv, Y., and Wang, F.-Y. (2021). Fii-centernet: an anchor-free detector with foreground attention for traffic object detection. *IEEE Transactions on Vehicular Technology*, 70(1):121–132.

Farrokhtala, A., Chen, Y., Hu, T., and Ye, S. (2018). Toward understanding hidden patterns in human mobility using wi-fi. In *2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)*, pages 1–4. IEEE.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645.

Feng, Y., Duives, D., Daamen, W., and Hoogendoorn, S. (2021). Data col-

lection methods for studying pedestrian behaviour: A systematic review. *Building and Environment*, 187:107329.

Foth, M., Choi, J. H.-j., and Satchell, C. (2011). Urban informatics. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 1–8.

Fruin, J. J. (1971). *Pedestrian planning and design*. Metropolitan Association of Urban Designers and Environmental Planners.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40.

Gehl, J. (1971). *Life Between Buildings*. Danish Architectural Press.

Gehl, J. (2010). *Cities for people*. Island press.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Gorrini, A., Messa, F., Ceccarelli, G., and Choubassi, R. (2021). Covid-19 pandemic and activity patterns in milan. wi-fi sensors and location-based data. *TeMA-Journal of Land Use, Mobility and Environment*, 14(2):211–226.

Gorrini, A., Vizzari, G., and Bandini, S. (2016). Age and group-driven pedestrian behaviour: from observations to simulations. *Collective Dynamics*, 1:1–16.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916.

Held, D., Thrun, S., and Savarese, S. (2016). Learning to track at 100 fps with deep regression networks. In *European conference on computer vision*, pages 749–765. Springer.

Hillier, B. (2012). The city as a socio-technical system: A spatial reformulation in the light of the levels problem and the parallel problem. *Digital urban modeling and simulation*, pages 24–48.

Hou, J., Chen, L., Zhang, E., Jia, H., and Long, Y. (2020). Quantifying the

usage of small public spaces using deep convolutional neural network. *PloS one*, 15(10):e0239390.

Ibrahim, M. R., Haworth, J., and Cheng, T. (2020). Understanding cities with machine eyes: A review of deep computer vision in urban analytics. *Cities*, 96:102481.

Idrissov, A. Y. (2012). A data cleaning framework for trajectory clustering. Master's thesis, University of Alberta, Edmonton, Alberta.

Jiang, K., Xie, T., Yan, R., Wen, X., Li, D., Jiang, H., Jiang, N., Feng, L., Duan, X., and Wang, J. (2022). An attention mechanism-improved yolov7 object detection algorithm for hemp duck count estimation. *Agriculture*, 12(10):1659.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., and Qu, R. (2019). A survey of deep learning-based object detection. *IEEE access*, 7:128837–128868.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

Kim, J. U., Park, S., and Ro, Y. M. (2021). Robust small-scale pedestrian detection with cued recall via memory learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3050–3059.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Leal-Taixé, L. (2014). Multiple object tracking with context awareness. *arXiv preprint arXiv:1411.7935*.

Li, J., Liang, X., Shen, S., Xu, T., Feng, J., and Yan, S. (2017). Scale-aware fast r-cnn for pedestrian detection. *IEEE transactions on Multimedia*, 20(4):985–996.

Lin, M. and Hsu, W.-J. (2014). Mining gps data for mobility patterns: A survey. *Pervasive and mobile computing*, 12:1–16.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. *International journal of computer vision*, 128:261–318.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Liu, Y., Sun, P., Wergeles, N., and Shang, Y. (2021). A survey and performance evaluation of deep learning methods for small object detection. *Expert Systems with Applications*, 172:114602.

Lorgna, L., Ceccarelli, G., Gorrini, A., and Ciavotta, M. (2023a). Video analytics for understanding pedestrian mobility patterns in public spaces: The case of milan. In *11th International Conference on Pedestrian and Evacuation Dynamics (PED2023)*, Eindhoven, The Netherlands.

Lorgna, L., Ceccarelli, G., Gorrini, A., and Ciavotta, M. (2023b). Video analytics for understanding pedestrian mobility patterns in public spaces: The case of milan. In *51th European Transport Conference 2023 (ETC 2023)*, Milan, Italy. Submitted for Publication.

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679.

Meinhardt, T., Kirillov, A., Leal-Taixe, L., and Feichtenhofer, C. (2022). Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8844–8854.

Messa, F., Ceccarelli, G., Gorrini, A., Presicce, D., Choubassi, and Choubassi (2022). Deep learning video analytics to assess vga measures and proxemic behaviour in public spaces. In *13th International Space Syntax Symposium (13SSS), 22-24 June 2022, Bergen (Norway)*, pages 1–22.

Nam, H. and Han, B. (2016). Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4293–4302.

Ning, G., Zhang, Z., Huang, C., Ren, X., Wang, H., Cai, C., and He, Z. (2017). Spatially supervised recurrent convolutional neural networks for visual object tracking. In *2017 IEEE international symposium on circuits and systems (ISCAS)*, pages 1–4. IEEE.

Niu, T., Qing, L., Han, L., Long, Y., Hou, J., Li, L., Tang, W., and Teng, Q. (2022). Small public space vitality analysis and evaluation based on human trajectory modeling using video data. *Building and Environment*, 225:109563.

Pang, Y., Cao, J., Wang, J., and Han, J. (2019). Jcs-net: Joint classification and super-resolution network for small-scale pedestrian detection in surveillance images. *IEEE Transactions on Information Forensics and Security*, 14(12):3322–3331.

Ravazzoli, E., Torricelli, G. P., et al. (2017). Urban mobility and public space. a challenge for the sustainable liveable city of the future. *The Journal of Public Space*, 2(2):37–50.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666.

Sindagi, V. A. and Patel, V. M. (2018). A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 107:3–16.

Solera, F. and Calderara, S. (2013). Social groups detection in crowd through shape-augmented structured learning. In *Image Analysis and Processing–ICIAP 2013: 17th International Conference, Naples, Italy, September 9-13, 2013. Proceedings, Part I 17*, pages 542–551. Springer.

Solera, F., Calderara, S., and Cucchiara, R. (2015). Socially constrained structural learning for groups detection in crowd. *IEEE transactions on pattern analysis and machine intelligence*, 38(5):995–1008.

Solera, F., Calderara, S., Ristani, E., Tomasi, C., and Cucchiara, R. (2016). Tracking social groups within and across cameras. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(3):441–453.

Sun, P., Cao, J., Jiang, Y., Zhang, R., Xie, E., Yuan, Z., Wang, C., and Luo, P. (2020). Transtrack: Multiple object tracking with transformer. *arXiv preprint arXiv:2012.15460*.

Tokmakov, P., Li, J., Burgard, W., and Gaidon, A. (2021). Learning to track with object permanence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10860–10869.

Tomè, D., Monti, F., Baroffio, L., Bondi, L., Tagliasacchi, M., and Tubaro, S. (2016). Deep convolutional neural networks for pedestrian detection. *Signal processing: image communication*, 47:482–489.

Tripathi, G., Singh, K., and Vishwakarma, D. K. (2019). Convolutional neural networks for crowd behaviour analysis: a survey. *The Visual Computer*, 35:753–776.

United Nations (2014). *World Urbanization Prospects, the 2014 Revision.* United Nations. Department of Economic and Social Affairs, Population Division.

Vanumu, L. D., Ramachandra Rao, K., and Tiwari, G. (2017). Fundamental

diagrams of pedestrian flow characteristics: A review. *European transport research review*, 9:1–13.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee.

Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., and Leibe, B. (2019). Mots: Multi-object tracking and segmentation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 7942–7951.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

Wang, Z., Zheng, L., Liu, Y., Li, Y., and Wang, S. (2020). Towards real-time multi-object tracking. In *European Conference on Computer Vision*, pages 107–122. Springer.

Wefering, F., Rupprecht, S., Bührmann, S., and Böhler-Baedeker, S. (2013). Guidelines. developing and implementing a sustainable urban mobility plan. In *Workshop*, page 117.

Whyte, W. H. et al. (1980). *The social life of small urban spaces*. The Conservation Foundation, Washington, DC.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE.

Wu, J., Zhou, C., Zhang, Q., Yang, M., and Yuan, J. (2020a). Self-mimic learning for small-scale pedestrian detection. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2012–2020.

Wu, X., Sahoo, D., and Hoi, S. C. (2020b). Recent advances in deep learning for object detection. *Neurocomputing*, 396:39–64.

Yu, Q., Luo, Y., Chen, C., and Chen, S. (2019). Trajectory similarity clustering based on multi-feature distance measurement. *Applied Intelligence*, 49:2315–2338.

Yu, X., Gong, Y., Jiang, N., Ye, Q., and Han, Z. (2020). Scale match for tiny person detection. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1257–1265.

Yuan, G., Sun, P., Zhao, J., Li, D., and Wang, C. (2017). A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, 47:123–144.

Zhang, S., Chi, C., Yao, Y., Lei, Z., and Li, S. Z. (2020). Bridging the gap between anchor-based and anchor-free detection via adaptive training sample

selection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9759–9768.

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2022). Bytetrack: Multi-object tracking by associating every detection box. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 1–21. Springer.

Zhang, Y., Wang, C., Wang, X., Zeng, W., and Liu, W. (2021). Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129:3069–3087.

Zhanga, F., Mirandaa, A. S., Duarte, F., Vale, L., Hack, G., Liu, Y., Batty, M., and Ratti, C. (2023). Urban visual intelligence: Studying cities with ai and street-level imagery. *arXiv preprint arXiv:2301.00580*.

Zhao, Y., Shi, F., Zhao, M., Zhang, W., and Chen, S. (2020). Detecting small scale pedestrians and anthropomorphic negative samples based on light-field imaging. *IEEE Access*, 8:105082–105093.

Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232.

Zheng, Y. (2015). Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):1–41.

Zhou, X., Koltun, V., and Krähenbühl, P. (2020). Tracking objects as points. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV*, pages 474–490. Springer.

Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. (2023). Object detection in 20 years: A survey. *Proceedings of the IEEE*.