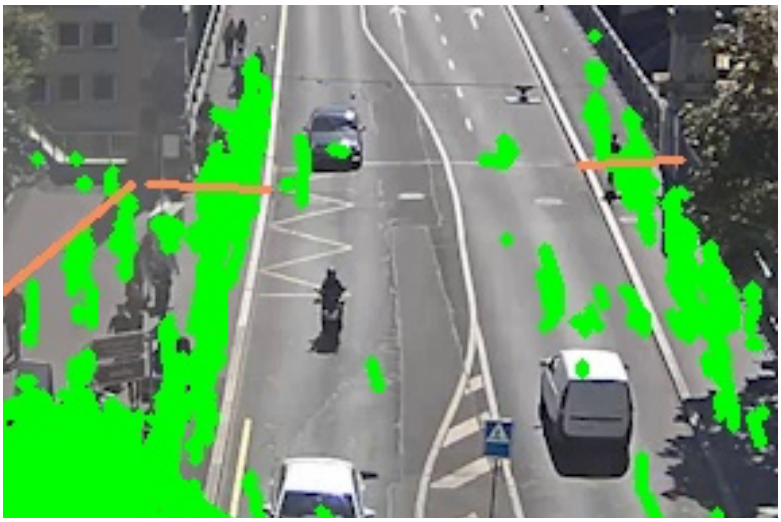
UNIL | Université de Lausanne
Faculté des géosciences
et de l'environnement

Master of Science in Geography

An advanced machine learning pipeline for pedestrian counting from video:
Performance comparison between different algorithms, model parameters and input sources

Max Henking

Sous la direction du Dr. Christian Kaiser
Et l'expertise de Raphaël Bubloz



Ville de Lausanne (2023) - analysed with SORT

Maîtrise universitaire ès sciences en géographie | Janvier - 2024

Abstract

Tracing and counting pedestrians is a growing area of interest in spatial analysis in the current context of fast growing and denser urban spaces. This research is part of this trend and presents an automated method that facilitates the extraction and analysis of pedestrian data from cameras for decision-support purposes in the field of spatial planning. There are two main focuses to this research: the first focuses on the extraction methods, which are tested by various combinations of algorithms and their associated parameters (detection threshold and tracking memory). The second assesses the reliability of these results by comparing algorithms and analysing raw images to validate these observations. The resulting optimised pipeline via the identification of algorithms, parameters and external factors improving pedestrian data extraction.

Keywords : Pedestrian counting, pedestrian detection, pedestrian tracking, machine learning, spatial planning

Résumé

Le suivi et le comptage des piétons est un domaine d'intérêt croissant en analyse spatiale dans le contexte actuel de croissance rapide et de densification des espaces urbains. Cette recherche s'inscrit dans cette tendance et présente une méthode automatisée qui facilite l'extraction et l'analyse des données sur les piétons à partir de caméras à des fins d'aide à la décision dans le domaine de la planification territoriale. Cette recherche s'articule autour de deux axes principaux : le premier se concentre sur les méthodes d'extraction, qui sont testées par diverses combinaisons d'algorithmes et leurs paramètres associés (seuil de détection et mémoire de suivi). Le deuxième évalue la fiabilité de ces résultats par comparaison entre algorithmes et analyse des images brutes pour valider ces observations. Il en résulte un processus optimisé via l'identification des algorithmes, paramètres et facteurs externes améliorant l'extraction de données piétonnes.

Mots-clés : Comptage de piétons, détection de piétons, suivi de piétons, apprentissage automatique, planification territoriale

Acknowledgements

With these few words, I would like to thank all the people who have accompanied and helped me throughout this master's thesis.

Firstly, I would like to thank Christian Kaiser, without whom this project would not have been possible. His suggestions and guidance on both the technical and methodological aspects have been of tremendous help throughout this work and allowed me to concentrate on the essential parts without taking too many detours.

My thanks to Raphaël Bubloz for agreeing to contribute his expertise to this work.

My sidekick Baptiste for the support, the constructive exchanges and the time spent out and about. Kerria, Romain and Loïc for their suggestions and review of this work. A special mention to Kerria and my family for their unconditional support and patience.

And finally, my thanks to all the anonymous contributors to the open source projects, forums and tutorials that enable the development of fantastic tools and make it easier to pass on knowledge to a wider audience.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and objectives

In the rapidly expanding landscape of urban environments, understanding the dynamics of pedestrian behaviour has become a critical issue for city planners and management. As cities continue to grow and addressing the majority of the world's population, accommodating the needs and behaviours of their inhabitants become imperative to ensure equitable access to the opportunities within these spaces. The United Nations (UN) identify four demographic megatrends exerting a profound influence on the demand for pedestrian-friendly urban areas. They are urbanisation, migration, population ageing and growth. Understanding and harnessing those trends is essential in order to create cities that are not only safer and more efficient but also sustainable in the long term (United Nations, 2022).

The first demographic megatrend is urbanisation. This socio-economic process can be observed across the globe by transforming former rural areas into urban ones and also shifting the spatial distribution of population from rural settlements to cities. This also includes significant shifts in occupational, cultural, lifestyle and general patterns. Thereby changing the traditional demographic and social structures of those spaces (Montgomery et al., 2013). This influx of people tends to create megacities and places immense pressure on city planners to design spaces capable of accommodating not only the rising population but also the multitude of transport flows navigating these urban areas.

The second demographic megatrend englobes all the implications related to migration. People are invariably drawn to urban settings by the promise of improved living conditions, ranging from enhanced professional opportunities to better access to healthcare and education. This attraction coupled with urbanisation translates into the emergence of megacities as major hubs of population and economic activities. This further explains the rise of urban dwellers and the need for good transport management.

The third demographic megatrend is the population ageing. This has far-reaching implications for the demand on high quality pedestrian infrastructure. It is necessary to take into account the special requirements for each age group in order to guarantee a safe and comfortable walking experience (actif-trafiC, 2022). This is done in order to ensure a fair access to urban facilities.

Lastly is the population growth. Even though the population growth is slower than in the 1950s. According to the United Nations Population Prospects 2019, the world population will reach 8.5 billion in 2030, 9.7 billion in 2050 and 10.7 billion in 2100. The slowing growth rate is mostly related to lowering rates of fertility and mortality. With the rising population and a predicted higher proportion of them living in urban environments (two-thirds by 2050), it is vital for cities to make accurate planning decisions.

In the centre of the 2030 agenda for sustainable development are seventeen development goals that were approved by all the members of the United Nations. The eleventh goal of the Sustainable Development Goals (SGDs) states that : «*Making cities sustainable means creating career and business opportunities, safe and affordable housing, and building resilient societies and economies. It involves investment in public transport, creating green public spaces, and improving urban planning and management in participatory and inclusive ways*» (United Nations, 2023). Sustainable transportation is one way of ensuring equal access to the benefits of living in dense urban areas (easier access to education, proximity of health facilities, economic opportunities, etc.). Also it can reduce the adverse effects that come with urbanisation such as urban sprawl, air/noise pollution and a highly congested street network.

In order to achieve those objectives, it is vital for planners to have access to reliable and precise data on multiple modes of transport. In the case of motorised vehicles, traffic impact assessments are mandatory for development review. In the case of pedestrian traffic, this type of data collection is missing. This study tries to create a pipeline to easily collect pedestrian data in order to provide useful insights into pedestrian behaviours and help decision makers in their work.

The demand for this type of data can be seen in multiple contexts. The first example is in train stations. The Swiss Federal Railways (SBB) have taken the initiative to standardise their passenger counting method to obtain more precise data regarding their customers' movements in train stations in order to enhance the traveller's experience and the railroad infrastructure management. Currently the collected data is limited to entries and exits in the stations which provides only a rough overview of the passenger movements. To more effectively respond to customer needs and optimise space planning in rail stations, it is necessary to go beyond such basic figures. The goal of their new system is to collect detailed data on the transit, stop and gathering areas for different types of customers. This can lead to a better planning of services, improvements to accessibility and ensuring a safe and pleasant travel experience for all travellers (SBB, 2023).

Another example of the use of precise data is in natural parks, where it is necessary to assess the potential impact of visitors on regional economy, capacity of roads, necessary infrastructure and the disturbance of wildlife. Those can only be estimated by using reliable data as in Rupf et al. (2008). In order to reduce the influence of the sensors on visitor's behaviour, they used acoustic slab sensors because

they are imperceptible and use little energy. This method lead to a high deviation between manually counted visitors and automatically counted visitors. As such, it is necessary to find an alternative making it possible to get reliable data in such contexts where the visitor's behaviour could be influenced by the presence of sensors too.

1.2 Research question

The first axis of this thesis centres on the critical task of obtaining reliable pedestrian data. To achieve this task, there currently exists a wide range of sensors. Bluetooth technology is utilised, leveraging signals emitted by pedestrians' devices to track their presence and movement within a specific area. Movement sensing technology can be used to capture and interpret the pedestrians' motion. Mobile signal tracking is another method of tracking using mobile phone signals to get pedestrian location. Lastly camera sensors offer a visual approach to pedestrian tracking.

In this study we will focus on cameras. In the contemporary urban landscape, they have emerged as a promising technology for tracking and monitoring pedestrian movements in various contexts. This technology offers the potential to capture rich and precise data, encompassing pedestrian traffic patterns, behaviour, and interactions, which is invaluable for urban planning and management.

Camera sensors, whether in the form of surveillance cameras, advanced imaging systems, or other sensor technologies, have the capacity to provide a wealth of information regarding pedestrian activities. Their deployment in urban environments holds the promise of collecting comprehensive, high-resolution data that can serve as a foundation for informed decision-making. The first research question would be : How to obtain reliable pedestrian data using camera sensors ? This axis of the thesis will delve into the method, techniques, and technologies involved in extracting pedestrian data using camera sensors.

The second axis of this thesis involves the analysis of the pedestrian data obtained through camera sensors and its subsequent application to specific contexts. Extracting data is only the initial step; the true value of this information lies in its analysis and its ability to offer insights and solutions within chosen urban contexts. This leads us to the second research question : How to analyse pedestrian data ? What can it be used for ? Through this axis, the research will focus on the critical phase of transforming raw data into meaningful information. The analysis will encompass various aspects, such as pedestrian flow patterns, congestion levels, safety assessments, and the impact on infrastructure. Furthermore, the thesis will explore how this analysed data can be applied to diverse settings, including public transportation hubs, natural parks, and urban centers.

The overarching objective of this thesis is to evaluate the reliability of camera sensors as a means of extracting actionable pedestrian information. This is achieved by assessing the accuracy, consistency, and completeness of the data acquired through

these sensors. The combination of these two axes forms a comprehensive framework to investigate the potential of camera sensors as a valuable tool for enhancing urban planning, safety, and efficiency in contemporary cities.

By addressing both the acquisition and analysis of pedestrian data, this research endeavours to contribute to the advancement of pedestrian-focused urban design and management, while also offering insights into the broader realm of sensor-based data collection and analysis for urban planning and decision-making.

1.3 Contributions

This thesis mainly contributes to the urban planning and camera sensor data analysis in the following ways :

- **Pedestrian data extraction** : this research provides a systematic review of pedestrian data acquisition and camera sensor technologies in the field of tracking and counting. By assessing the reliability of those technologies, this work offers practical insights into the deployment and configuration of camera sensors to ensure reliable pedestrian data.
- **Contextual applications** : this work also applies beyond theory and applies the extracted data to real-world scenarios. By doing so in various settings, it also shows the practical utility behind extracting data from camera sensors. It also offers valuable insights for decision-makers.
- **Sustainable urban planning** : the study aligns with the eleventh goal of SGD's in addressing the need for sustainable urban development and efficient public spaces. It alights the use of reliable pedestrian data in planning pedestrian friendly urban infrastructure, reducing congestion, improving safety and contributing to sustainable planning decisions.

1.4 Expectations

This research encompasses two primary objectives, each of which contributes to our understanding and practical utilisation of pedestrian data from various sources :

Pedestrian information extraction : The expected results of this thesis are to obtain relevant information related to pedestrian flows regardless of the source of the video and and its context. Moreover, it is to provide a simple and reproducible framework for the extraction of pedestrian flows using diverse video sources (webcams, camera sensors, etc).

Pedestrian information analysis : The analysis of the obtained pedestrian flows conducted to offer easily interpretable information in order to provide data for

urban planning decisions. It is expected of this thesis to provide origin destination flows and give a reliable way to obtain pedestrian data.

1.5 Thesis structure

The following structure will be used to answer these research axes. Section 2 will be a literature review of the two axes. Section 3 will describe the data and study sites used for this thesis. Section 4 will review the methodology for the count of pedestrian and for the analysis. Section 5 will present the different results and findings. Section 6 will expand on the limits and further improvements. In section 7, the findings will be synthesised.

Chapter 2

Context and case study

In this section, the state of the art for the two axes of this study will be contextualised. This part was accomplished using Google Scholar and Papers With Code in order to find relevant previous research.

In the first part, the data collection method will be discussed. As previously stated by Hollenhorst (1992), this part is often very time-consuming and resource consuming. It can further be categorised into three main categories : self-counting, direct-counting and indirect counting methods. This study will focus on indirect counting methods in the domain of computer vision. Computer vision is a field of machine learning whose objective is to learn from images and extract useful information from those images. In a way, its objective is to let a computer see, observe and understand just like the human eye. For pedestrian tracking, computer vision studies have mostly separated this problem into three main sub tasks : image classification, object detection and object tracking. In order to get a simple overview of the state of the art on this subject, this structure will be used in the following chapter.

In the second part, we will expand on the multiple studies made on assessing pedestrian data and how they used such data for effective planning and what indicators are used to make accurate predictions. As such, this section will focus on the practical implementations and uses of pedestrian data in the field of geotourism, geomarketing, urban planning and crowd monitoring.

Definitions

- Computer vision : It is a sub-field of Machine Learning whose main objective is to let machines achieve human-like performance on interpreting and understanding visual data. In recent years, the main sub tasks associated with computer vision are the segmentation, classification and detection of objects in images or videos (Martin, 2020).

- Neural Networks : Also known as Artificial Neural Networks (ANN), they are inspired by the neural network of the human brain and try to mimic its function. A basic neural network is composed of an input layer, multiple hidden layers and an output layer (Géron, 2022).

- Convolutional Neural Networks (CNN) : Similar to ANN's, the CNN's emerged

from the study of the visual cortex of the human brain. These neural networks take as input an image or video and extract information from that. One important building block of such neural networks is the convolutional layer. This layer acts as a filter to extract information by from images (Géron, 2022).

- Image classification: The task of labelling and categorising groups of pixels or vectors in an image. This is done based on predefined characteristics such as colours or textural elements (Shinozuka and Mansouri, 2009).
- Object detection: The task of object detection builds upon the classification results to assign a location and size to the objects of interest (Redmon et al., 2016).
- Object tracking: A task in computer vision that can be partitioned into locating objects, re identifying detected objects across frames and calculating the individual trajectories based on an input video (Luo et al., 2021).

2.1 Pedestrian tracking

This section focuses on the three main sections of a pedestrian data extraction pipeline. The first section reviews the evolution of the image classification methods, the second of the object detection methods and the third reviews the literature around object tracking.

Image classification

Image classification is a crucial step in pedestrian tracking, its main objective is to identify pedestrians within a scene captured by cameras. Over the years, multiple methods have been developed in order to accurately complete this task.

In early works on pedestrian tracking, image classification was done using traditional image processing techniques and by extracting features on each pixel of the input image. A lot of work was done in order to provide accurate ways of representation. As such, there is an extensive literature on object classification as a part of the object detection problem. A few of the relevant articles will be mentioned here. These articles all were written in response to a growing need for accurate pedestrian data acquisition using camera sources various industries including surveillance, autonomous driving, recognition systems or even market research. The work of Papageorgiou and Poggio (2000) introduce a general system for object detection in clustered, free flowing spaces. In order to provide an accurate representation of pedestrians, the main objective is to have high inter-class variability and low intra-class variability. To guarantee that, the extracted features for pedestrians must be consistent for every pedestrian and very distinguishable from the other classes. Their contribution is the use of Haar-like features to help the computer

detect pedestrians. Those features called Haar Wavelets are the result of the decomposition of an image into local oriented intensity differences between adjacent regions. This gives a way to compare complex structures while removing noise and lets the machine classify the image between objects of interest or not. Dalal and Triggs (2005) introduce another method called Histograms of Oriented Gradients (HOG) to classify an image. This method is based on the idea that the appearance and shape of objects can be described by using local intensity gradients without knowing their exact position. Their objective is to provide a efficient system. It was done by following the steps pictured in Fig. 2.1. The HOG's contrast-normalise the gradient histograms in order to reduce the impact of illumination, shadowing and overall noise in the input image.

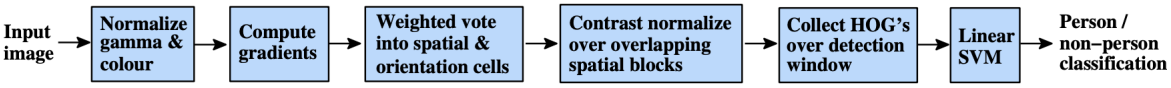


Figure 2.1: Structure for the image classification using the HOG method (Dalal and Triggs, 2005)

Yoshinaga et al. (2009) use more traditional image processing techniques in their paper on people counting. The first is the background subtraction to classify background pixels from non-background ones ([b] in figure 2.2). This is done using a Parzen density estimation algorithm. Shadows and elements subject to variations related to illumination also stay after the first step. As such, the next processing step is the removal of shadows because it may affect overall performance and results ([c] in figure 2.2). This method is done by applying a YUV colour filter that identifies the a shadow as a slight variation of colour of the original object.

(a) Input image (b) Subtracting the background (c) Eliminating the shadow regions

Figure 2.2: Segmentation sequence in blob descriptor technique (Yoshinaga et al., 2009)

Recent advancements have shifted to use convolutional neural networks (CNN's) for the image classification step too. One notable contribution is the work by Redmon et al. (2016) on the You Only Look Once (YOLO) algorithm. This solution is a real-time classification and detection algorithm that is still vastly used in various pedestrian tracking applications. The classification is done by separating the image into multiple smaller grid cells where each of them is given bounding boxes and a confidence score. This confidence score represents the trust of the model in the

presence or absence of an object and how accurate the associated bounding box is. Similar to CNN's other approaches such as the sliding window where a classifier is slid by a regular step over the entire image to predict the class of each pixel (Felzenszwalb et al., 2010). Another one like the R-CNN use region proposal methods to generate potential bounding boxes and then run a classifier on those bounding boxes (Girshick et al., 2014).

Transfer learning has also played a pivotal role in the advancement of image classification for pedestrian tracking. In a recent work by Han et al. (2022), pretrained VGG-16 and Feature Pyramid Network (FPN) were used in order to improve the accuracy of the classification. This is also done in order to reduce the need for vast amounts of labelled training data.

A large panel of tools is available for experts to use in order to classify an image. Each of these methods obtains varying results in terms of accuracy, cost or execution time. Even then, it is possible to understand that the task of classification is a crucial one for our study in order to reduce the amount of resources needed by a computer to identify pedestrians on video images.

Object detection

The second part of pedestrian tracking involves the correct detection of pedestrians within a given image and video frame. Object detection is a field of computer vision whose main objective is to detect the boundaries and position of objects in an image or video. The detection methods have, as with image classification, evolved from traditional techniques to deep-learning approaches.

Early works were based on traditional image processing techniques. Viola and Jones (2001) introduce a method using previously classified images and their features to detect objects of interest. This is done by using multiple classifiers in order to reduce the number of necessary features needed to find the pedestrians. In the work of Yoshinaga et al. (2009), object detection is done by extracting features on the previously classified frames. Once the features match a certain threshold they are detected to be an object of interest (e.g.: pedestrians).

With the advent and the growing use of deep-learning techniques, multiple breakthroughs in object detection were achieved. According to the Papers with Code website ("Papers with Code - Object Detection", 2023), the current state of the art methods can be classified into two main categories which are :

- One-stage methods like Overfeat, YOLO and SSD
- Two-stage methods such as the Faster R-CNN

One stage methods are called as such because of the pipeline used by the models. They first hypothesise bounding boxes, then extract pixels or features for each box and apply a high-quality classifier to detect the objects of interest (Liu et al., 2016). In most of those models, varying methods are used to extract features from an image

such as sliding windows or region proposals. Those features are then used to localise and characterise the objects detected. Sermanet et al. (2014) introduce Overfeat, an algorithm that applies a Convolutional Neural Network on a sliding window to execute the classification and detection steps in a single pipeline. As presented before, YOLO is a one-stage method whose main objective is to look at an image once and detect objects in the image. It simplifies the detection problem into a regression one and differs from its sliding window counterparts because it takes the whole image into account when performing the classification and detection steps (Fig. 2.3). Single shot Multibox Detector (SSD) (Liu et al., 2016) is a single stage method that is more accurate than YOLO and almost as accurate as the Faster R-CNN one. This is partly due to deleting the steps of hypothesising bounding boxes and resampling features. Also it makes small improvements by adding multiple filters to perform detection at multiple scales.

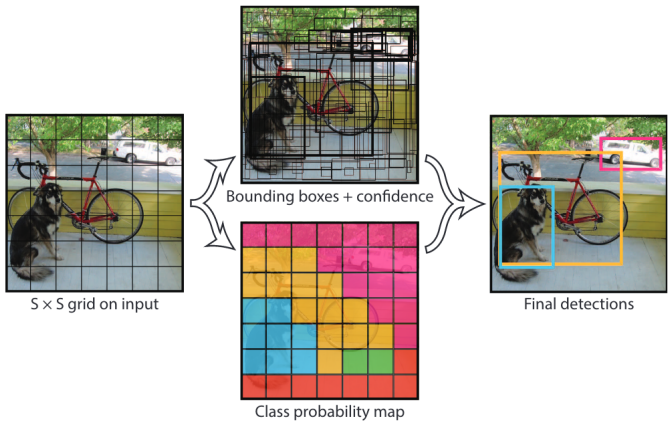


Figure 2.3: YOLO model (Redmon et al., 2016)

Two-stage methods have the same first step as their one-stage counterpart but the second stage often consists of refining the detections to improve the accuracy and obtain better results. Those methods often come with an increase in the computational cost and the reduction of execution speed. As stated by Liu et al. (2016), Faster R-CNN has the highest accuracy on state of the art datasets such as PASCAL VOC but they are much slower (7 frames per second) and cannot perform on real-time video feeds. Faster R-CNN uses a Region Proposal Network (RPN) in the first step and an R-CNN for the second step.

Object tracking

Lastly, the detected objects are tracked across multiple frames in order to identify their patterns. This can be understood as a data association task and has been the subject of a lot of research due to the various challenges brought about by this task such as occlusion, varying velocities, environment changes (illumination, weather, etc.) and other abrupt changes.

Traditional approaches like the Kalman filter have been widely used in pedestrian

tracking. It is an iterative mathematical process that uses a set of equations and consecutive data inputs to quickly estimate the true value, position, velocity, etc. of the pedestrian being detected (Kalman, 1960). But those approaches are prone to errors due to aforementioned challenges. This led to the development of more sophisticated approaches such as SORT (Bewley et al., 2016) who complemented the Kalman filter with the Hungarian algorithm to make more optimal decisions. Or DeepSORT (Wojke et al., 2017) who integrates deep-learning elements in the tracking process in order to improve accuracy and quality of the tracked pedestrians. According to Luo et al. (2021), most Multi Object Tracking (MOT) works can be grouped under two main axes (Fig. 2.1). The first axis is detection-based tracking (DBT) whose first step is to detect the objects of interest in a sequence of images. Afterwards, the trajectory hypothesis is extracted and the sequence is linked to create a trajectory for each object. The second axis is detection-free tracking (DFT), this approach relies on a manual initialisation of the objects of interest in the first frame. Its objective is then to locate the objects in the subsequent frames.

	DBT	DFT
Initialisation	automatic, imperfect	manual, perfect
n° of objects	varying	fixed
Applications	specific type of objects	any type of objects
Advantages	can handle varying number of objects	free of object detector
Drawbacks	performance depends on object detection	manual initialisation

Table 2.1: Comparison between DBT and DFT (Luo et al., 2021)

In a recent paper, another approach to the MOT problem has been identified. The tracking-by-counting approach has been researched as a way to bridge the gap between counting, detection and multi-object tracking. This was done in order to answer the need of a reliable tracking model in crowded scenes. As seen in figure 2.4, Ren et al. (2021) propose an improved version of the MOT problem. It is still based on the same two parts : object detection and data association. But in the case of their study, they improved the method by incorporating constraints to increase performance. The first constraint is a count constraint. This constraint is based on density maps extracted on each frame of the sequence. This leads to more precise counts in the object detection step. The constraint used for data association is a flow constraint that is also extracted from the density maps to create a graph that reduces the movement probabilities of pedestrians to a few hypotheses.

Synthesis of the literature on pedestrian tracking

The integration of those three steps form the backbone of an effective pedestrian tracking application using webcams. The previously reviewed literature highlights the significant shift from traditional approaches to deep-learning based ones in recent years. As the field of computer vision is ever changing and very vast, this chapter is only a partial glance at the many algorithms and possibilities related to pedestrian

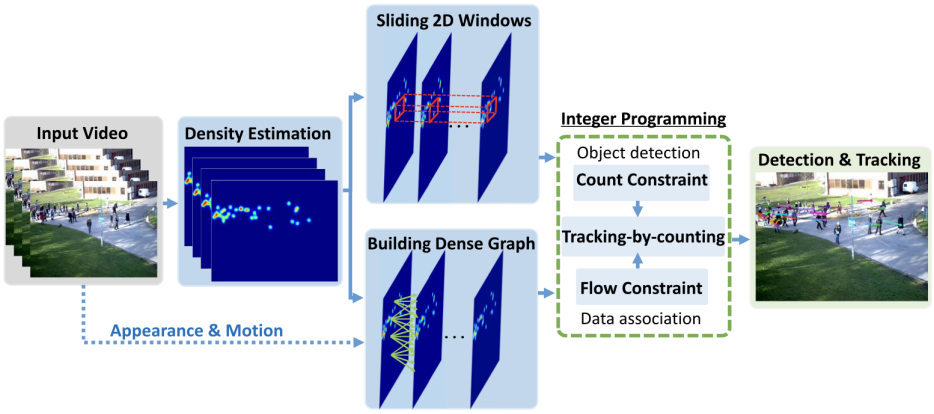


Figure 2.4: Tracking by counting (Ren et al., 2021)

tracking using video sources. This study will build upon this foundation to try and propose a simple pipeline for planners to acquire reliable data using those methods.

2.2 Pedestrian data analysis

In this part, we focus on the practical use cases for pedestrian data and how this type of data is used in the multiple fields.

Pedestrian data for geomarketing

Pedestrian data plays a pivotal role in enhancing marketing strategies by providing valuable insights into consumer behaviour and preferences. Retailers and marketers can utilise this data to analyse foot traffic patterns, understand the flow of people in various locations and identify popular areas for potential storefronts or advertising placements (Baviera-Puig et al., 2016). Geomarketing strategies can be further optimised by tailoring promotions based on the demographics and behaviours of pedestrians in particular areas, ultimately improving the targeting precision of marketing efforts. Moreover, pedestrian data aids in site selection for businesses, enabling them to choose locations that align with their target audience's habits and preferences. Overall, the applications of pedestrian data in geomarketing empower businesses to make informed decisions, enhance customer engagement, and maximise the impact of their marketing initiatives (CARTO, 2024).

Pedestrian data for geotourism

The applications of pedestrian data in geotourism are extensive and contribute significantly to enhancing the overall tourist experience. Pedestrian data allows for the analysis of visitor movement patterns within specific geographical locations, enabling tourism planners and operators to understand popular routes, attractions, and points of interest. By leveraging this data, destinations can optimise their infrastructure, improve accessibility, and create more engaging and personalised expe-

riences for tourists (Rupf et al., 2006). Geotourism businesses can utilise pedestrian data to design interactive and location-specific mobile applications, offering tourists real-time information, guided tours, and recommendations based on their preferences and historic movement (Ribeiro et al., 2018). Additionally, insights derived from pedestrian data can aid in the development of sustainable tourism practices by identifying areas prone to congestion or overuse. This information empowers destination managers to implement effective crowd management strategies, preserving natural and cultural resources while ensuring a positive and enjoyable experience for visitors. Ultimately, the applications of pedestrian data in geotourism contribute to a more informed, sustainable, and immersive travel environment (Cessford and Muhar, 2003).

Pedestrian data for urban planning

Pedestrian data plays a crucial role in modern urban planning, offering valuable insights that contribute to creating safer, more efficient, and people-centric cities (Gehl, 2013). By analysing pedestrian movement patterns, city planners can optimise infrastructure development, such as the placement of crosswalks, pedestrian-friendly zones, and public transportation hubs. Understanding foot traffic helps in identifying areas with high pedestrian demand, allowing for the strategic allocation of resources and the implementation of measures to enhance overall accessibility (van der Spek, 2009). Pedestrian data is instrumental in designing urban spaces that prioritise walkability and promote active lifestyles, fostering a sense of community and reducing reliance on private vehicles. In order to do so, Sevtsuk (2021) simulates the pedestrian flows in Cambridge. He then proposes a similar traffic impact assessment to evaluate the impact of pedestrian traffic on the level of service in the field of study. Additionally, planners can use this data to assess the impact of new developments or urban interventions on pedestrian flow, ensuring that the changes align with the needs and behaviours of the community. Overall, the applications of pedestrian data in urban planning contribute to the creation of more sustainable, liveable, and inclusive cities (Nieuwenhuijsen, 2020).

Chapter 3

Data

In this chapter, the data used in the study will be presented. Steps in the acquisition that were not considered in the Methodology chapter will be introduced such as the use of software to download videos from the web or the hardware used for the capture of videos.

3.1 Lausanne livestream video

Figure 3.1: Point of view of the camera (Ville de Lausanne, 2020)

The Bessières bridge is located in the City of Lausanne. It is located in the vicinity of multiple points of interests (the high school of la Cité, the cathedral of Lausanne). The webcam is located on top of the Pyxis (House of numerical culture and exploration) building. It is centered on the bridge and can therefore see all the traffic flows crossing the bridge. The video live stream is available on YouTube on the official channel of the city of Lausanne[1].

In order to make use of this video feed, we used the youtube_dl application. YouTube-dl is a command-line application that helps the user download videos directly from YouTube. It is released under the public domain by the copyright owners and can therefore be used and modified by each user. The following procedure has to be done in order to use the application. It is done in the terminal and assumes that Python (2.6, 2.7 or 3.2<) is already installed on the computer :

[1]Link to the livestream: https://www.youtube.com/watch?v=y3sMI1HtZfE

- Clone the original GitHub repository to the working directory :

```
git clone https://github.com/ytdl-org/youtube-dl.git
```

- Change the current directory to the cloned youtube-dl repository :

```
cd path/to/youtube-dl
```

- Download the video :

```
python -m youtube-dl -f 96 [URL to the youtube video stream]
```

In some cases, some parameters may have to be specified such as the ffmpeg-location. In case an error crops up following the preceding procedure, the README.md file on the GitHub repository introduces the optional parameters that can be specified.

Image specifications

Parameters	
Date	01.09.2023
Start hour	12:05
Length	5:59:15
Size of file	2.7 Go
Dimensions	1920x1020
Frames per second	30

(a) First video

Parameters	
Date	17.10.2023
Start hour	16:58
Length	5:59:29
Size of file	2.7 Go
Dimensions	1920x1020
Frames per second	30

(b) Second video

Table 3.1: Parameters for the videos used in the study

Table 3.1 gives us general information regarding the videos used for this study. Figure 3.2 is a schema of the scene with the different traffic axes and the position of the camera.

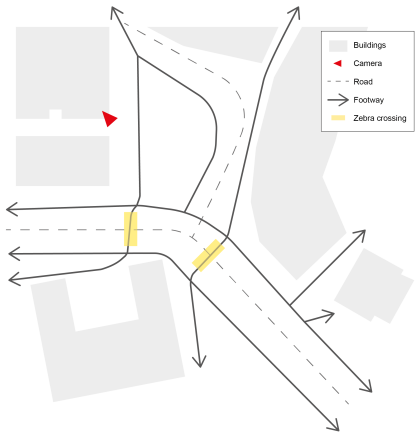


Figure 3.2: Schema of the camera position on the Bessières bridge

3.2 Lavaux images

This is part of a bigger counting project done for "Lavaux Tourisme". It is mandatory for regions such as the Lavaux Region to count the traffic flows in order to keep being registered in the UNESCO World Heritage List. As such, the project is to quantify the affluence in the region along a specified itinerary and using multiple cameras and Bluetooth sensors. The data that was to be collected in this location was not readily available in time for the completion of this thesis. As such, this section shows a potential application of the developed pipeline with the associated installation and hardware. The counting segment localisation is between the villages of Epesses and Rivaz. Those are multi use road segments which can be taken by foot, motorcycles or even cars. The camera is positioned on the chemin du Calamin. Figure 3.3 is a schema that shows the position and installation of the camera on the chemin du Calamin. It is installed with a solar panel and a battery to be able to guarantee a long period of detection without the need to be connected to the electric grid. In Fig. 3.4 we can see the solar panel and the camera attached to the fence, the black box contains the battery. As such, the hardware for this type of installation is made of few components and can be installed in a wide range of places.

Image specifications

As mentioned above, due to some technical difficulties and the time constraints of this study, no videos were taken at this spot.

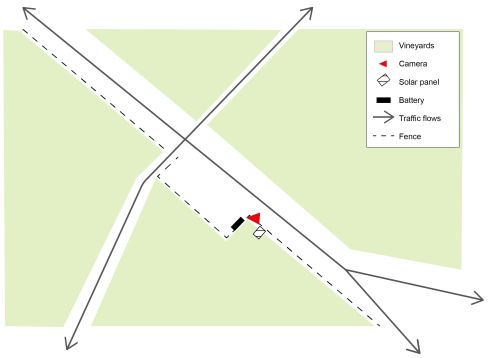


Figure 3.3: Schema of camera position on the chemin du Calamin

Figure 3.4: Camera, battery and solar panel (Christian Kaiser, 2023)

Chapter 4

Methodology

This chapter explains the various steps needed for the use of a pedestrian tracking system based on video sources. It is composed of three main parts : the pedestrian detection, the pedestrian tracking and the pedestrian data analysis. Its main objective is to clarify some key notions, explain the theoretical basis and present the tools used for the conception of the final product. It shows the various theoretical elements used for the detection, tracking and counting parts of our study. It also provides all relevant resources needed to reproduce the steps on other datasets or videos. All the code snippets provided in this chapter are written in the Python language if not specified otherwise. The complete scripts can be found at : https://github.com/MaxHenk/pedestrian_counting.

4.1 Pedestrian detection

For the goal of this study, we decided to divide the detecting and tracking in order to reduce the need for data storage in mobile systems such as seen on Figure 3.4. This is also done in order to let the system run on longer time frames without the need for too much human intervention. Another reason for the separation is to reduce the hardware needed for storing the detections and removing the need for processing power allotted to tracking. First the theoretical basis will be presented and then the actual implementation.

4.1.1 Theoretical basis

For the classification and detection part of our work, we used a MobileNetV2-SSD based method. This method is a lightweight model whose main purpose is to be executed on mobile and embedded applications. This class of models was introduced by Howard et al. (2017) as a direct response to the rising need of efficient solutions to use in diverse real-world applications such as autonomous driving cars, augmented reality or in wider uses such as pedestrian localisation.

The architecture of this model is constituted of multiple layers (Table A.1 in the appendix shows the exact structure of the MobileNet model). The first layer is a traditional convolutional that takes a $D_i \times D_i \times M$ size image as input (Figure 4.1). It outputs a $D_i \times D_i \times N$ image where D_i is the width and height of the image, M is the number of input channels and N is the number of output channels.

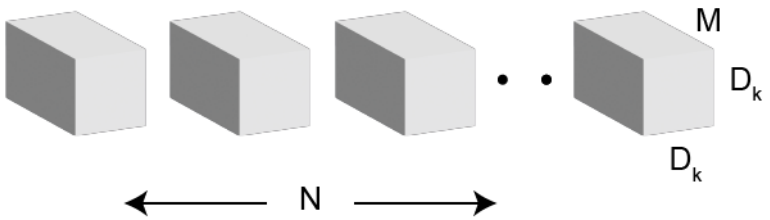


Figure 4.1: Filters used for traditional convolution layers (modified from Howard et al., 2017)

The core layer used in the model is a depthwise-separable convolution layer (Figure 4.2). This is actually constituted of two layers, a depthwise convolution (noted Conv dw in table A.1) and a pointwise convolution (Figure 4.3). The depthwise convolution is a convolution layer of size $D_k \times D_k \times 1$ and it has the same number of filters as input channels (M).

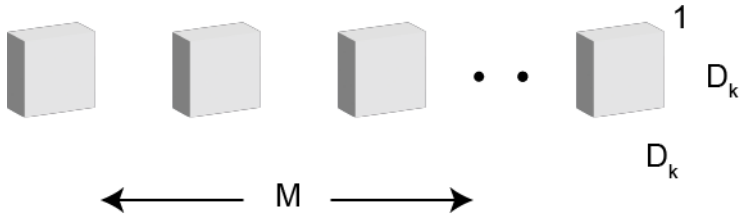


Figure 4.2: Filter used for depthwise convolution (modified from Howard et al., 2017)

The result of the depthwise convolution layer is a stack of channels that does not take into account intra-channel information. In order to get that, a linear combination of the output of the depth layer is done. The pointwise convolution of size $1 \times 1 \times M$ is used to extract new features.

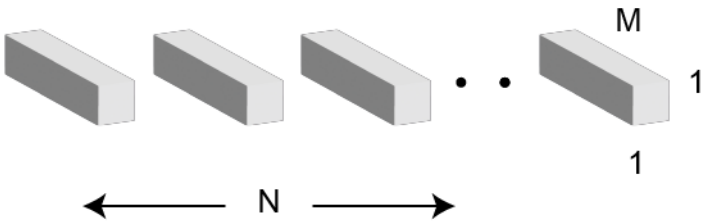


Figure 4.3: Filter used for pointwise convolution (modified from Howard et al., 2017)

The main advantage of this layer is in the improvement of computational cost compared to standard convolutional layers. Standard convolutional layers have a cost of :

$$D_i \times D_i \times M \times N \times D_k \times D_k \tag{4.1}$$

In the case of depthwise convolution layers, the computational cost is :

$$D_i \times D_i \times M \times D_k \times D_k + M \times N \times D_i \times D_i \tag{4.2}$$

Howard et al. (2017) state that this type of layer uses 8 to 9 times less computation

and is therefore very useful for mobile applications such as the one needed for this study. The model was trained on the COCO dataset (Lin et al., 2015). MobileNet was trained on a wide range of objects (Table A.2 shows all the class names and their class ids).

The other detection model is the PedNet, it is based on the DetectNet architecture (Fig. 4.4) and has the same functioning principle. Its main objective is that, given an input image and an associated grid square. The model must predict whether an object is present and the location of the bounding box corners for that object. This implies that this model only detect one type of object which, for PedNet, is pedestrians. The architecture associated with this label is pictured in Figure 4.4. Its first main component is a fully-convolutional network (FCN) similar to GoogleNet but without the input layers, final pooling and output layers (Szegedy et al., 2014). This model does not use fully-connected layers in order to accept varying sizes of image inputs and also to use the pretrained GoogleNet model. The second step is by clustering and filtering the bounding boxes based on a preset threshold. Lastly a confidence score is attributed to each bounding box.

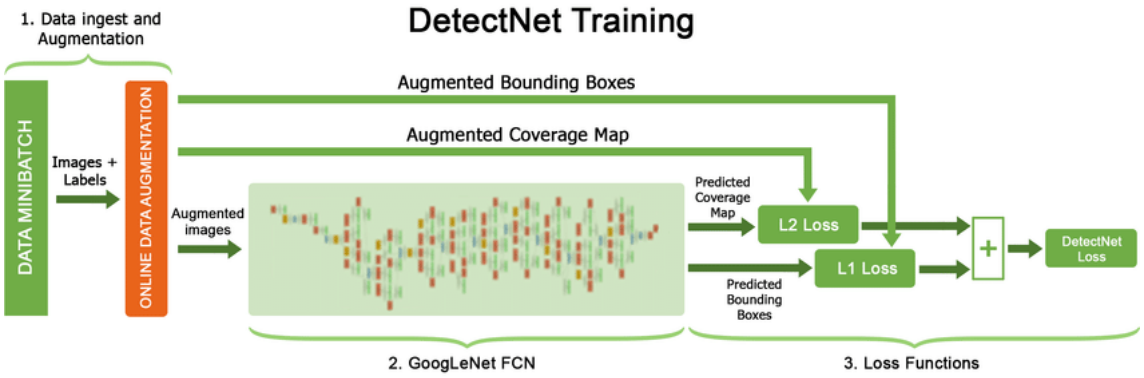


Figure 4.4: DetectNet architecture for training (Tao et al., 2016)

4.1.2 Implementation

The detection part of our study was done on a Jetson Nano Developper Kit. This machine is «*[...] a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts*» (NVIDIA, 2023). The first part of the deployment needs the Jetson Nano to be connected to the Internet to download the jetson-inference repository. It was created by NVIDIA to give the tools to implement Deep Neural Networks (DNN) on Jetson devices such as the Jetson nano developper kit. The DNN documented in this repository are :

- imageNet : image classification algorithms such as AlexNet, GoogleNet or ResNet. Those output class probabilities over the whole image.

- detectNet : object detection algorithms that focus on extracting bounding boxes of multiple objects in the frame.
- segNet : semantic segmentation algorithms, similar to its image classification counterpart but instead of focusing on the whole image, the focus is on each assigning a class to each pixel.
- poseNet : pose estimation algorithms focus on recreating a skeleton to objects. It focuses on detecting movements such as hand gestures.
- actionNet : action recognition algorithms use video frames to recognise a certain action/activity. The backbone of such algorithms is often image classification algorithms.

In this part of the study, we will focus on the detectNet. This type of algorithm takes an image or video file as input and outputs a text file containing the detected objects. The pre-trained detection algorithms was trained on 300x300 images, therefore it is necessary to preprocess the videos to match those dimensions in order to maximise the accuracy of our detections and so that the algorithm doesn't reshape the input images. If not done, the results will be distorted such as seen in Figure 4.5 where the tracks crossing the pedestrian passage are slightly under the expected area.

Figure 4.5: Tracks (in green) with a 0.5 detection threshold on a non cropped video

As such, to use the pretrained models provided in the original GitHub repository, it is necessary to first crop the videos in a 300x300 format so the coordinates of the tracks are not distorted and can be adjusted to the accurate coordinates. To do so, a video processing software can be used in order to extract a cropped version of the video. To further test our tracking and counting pipeline, we vary the preprocessing of the input videos (the first frames of those videos can be seen in appendix A.3). Since the size of the image is fixed, the difference in preprocessing lies in the difference in scale. In videos S2 (A.2b) and O2 (A.2d), the pedestrians are "bigger" than in videos S1 (A.2a) and O1 (A.2c).

The following line is used to launch the detection script on the terminal of the Jetson device and can be further modified by the user depending on specific parameters.

```
python3 /path/to/detection/script.py /path/to/video/file
        /path/to/output/folder
```

In Table 4.1 are the parameters that were changed while launching the detection script for this study with a short description and their potential values.

key	description
–network	This flag is followed by the name of the network to use, by default it uses the ssd-mobilenet-v2 but it can take any preloaded models that are installed.
–threshold	This flag is the minimum detection threshold, it is the minimum confidence score a detection has to have in order to be considered and saved to the outputs. The value ranges between 0 and 1.

Table 4.1: Input parameters for the detection

For this study, the algorithms used for detection were the MobileNet and MobileNet inspired PedNet algorithms. These were chosen because of similarities between the training datasets and the videos used in this study. This was done in order to maximise the accuracy of our detections. The detection threshold was chosen by first comparing the visual tracks obtained by applying a threshold value of 0.25, 0.3, 0.5, 0.75 (see appendix A.5). Based on those results and after discussion with the supervisor of this thesis, the chosen thresholds were 0.3 and 0.5.

For each detection, the following informations were written to a text file :

$$[confidence\ score, x1, y1, x2, y2, frame\ number, class\ id, date, hour] \tag{4.3}$$

The confidence score reflects the certainty by which the computer assigns a class to the detected bounding box. The x1, y1, x2, y2 are the coordinates of the bounding box. The frame number equals to the count of frames since the start of the video. The meaning of the class id changes depending on the algorithm used (the labels related to MobileNet are found in the appendix A.2). The date and hour of the detection are extracted using the python library *datetime* but in order to have accurate information, an additional component or a stable internet connection needs to be installed on the Jetson nano developer kit. In order to obtain a video stream, it is also needed to connect a camera module to the device. In the case of this study, videos extracted from the Internet were used instead of a camera stream. Also the code was designed to generate a new text file every hour, this was done in order to ensure that the text files created by the detection script (*my_detection.py* available on GitHub) never got too big in size which could result in corrupted files. Those hourly files are then merged together once they are extracted and ready for the next

section. This merging of files in the same directory can be done using the following shell command :

```
cat *.txt > detection_outputs.txt
```

This results in a detection_outputs.txt file that can be used in the following sections to get information about the whole duration of capture.

4.2 Pedestrian tracking

This part of the study can be done on a standard computer and it is therefore less relevant to reduce the computational cost of the scripts or the storage options, contrary to the detection part. This differentiation of the steps makes it possible for the end user to easily switch the tracking algorithm. It comprises of a theoretical section further explaining the model used and the exact methods associated with it. The second part explains the implementation of this model and focuses on the practical requirements needed for the use of said model.

Libraries used

All the scripts were written in Python and use the following libraries :

- cv2 : Also known as OpenCV, it is an open source library widely used for computer vision and machine learning projects. It contains a wide variety of both traditional and state of the art algorithms.
- numpy : It is an open source library mainly used for numerical computing in Python.
- sort : The sort library is the result of the work by Bewley (2016) and lets the user implement the SORT algorithm in a simple way.
- os : A built-in Python library used to interact with the operating system.

4.2.1 Theoretical basis

The tracking was done using the SORT (Simple Online and Realtime Tracking) algorithm (Bewley et al., 2016). In order to accurately track a pedestrian's identity between frames, each detected pedestrian is attributed multiple features in the form of :

$$pedestrian = [x, y, s, r, \dot{x}, \dot{y}, \dot{s}] \tag{4.4}$$

where x and y represent the horizontal and vertical positions of the detected pedestrian, while s and r represent the scale and the aspect ratio of the bounding box. The $\dot{x}$, $\dot{y}$ and $\dot{s}$ parameters are the associated velocities to these parameters. The aspect ratio is the only constant and therefore has no velocity associated to it. If a pedestrian is already detected, the associated bounding box is updated and the

velocity components are then dynamically updated using a Kalman filter. Figure 4.6 is a visual representation of the iterative process behind this filter.

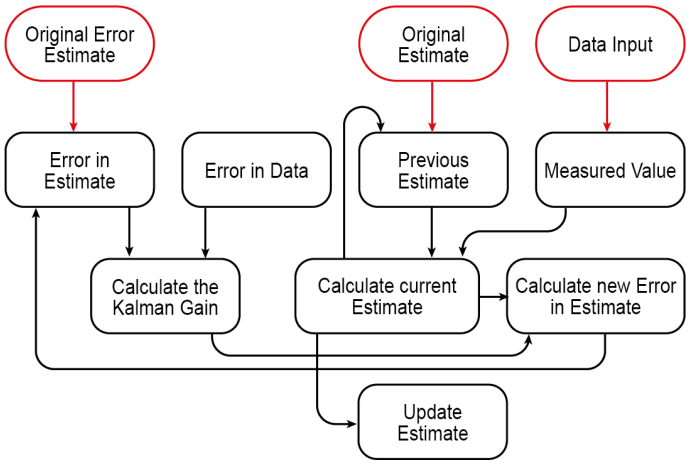


Figure 4.6: Diagram of the Kalman filter (Michel van Biezen, 2015)

As shown in Figure 4.6, the whole filter revolves around three formulas to calculate the Kalman gain, the current estimate and the new error in the estimate. Considering the following table :

Kalman Gain	KG	Current estimate	Est_t
Error in estimate	E_{Est}	Previous estimate	Est_{t-1}
Error in measurement	E_{Mea}	Measurement	Mea

Table 4.2: Nomenclature for the Kalman filter (modified from Michel van Biezen, 2015)

The Kalman gain varies between the values of 0 and 1. The closer its value is to 1, the more accurate the measurements are and the more unstable the estimates are. It is a way to quantify how much of the new measurement (Mea) is useful for the current estimate (EST_t). The three following formulas are the mathematical basis for the calculation of the Kalman filter.

$$KG = \frac{E_{Est}}{E_{Est} + E_{Mea}} \tag{4.5}$$

$$Est_t = Est_{t-1} + KG[Mea - Est_{t-1}] \tag{4.6}$$

$$E_{Est_t} = \frac{E_{Mea}E_{Est_{t-1}}}{E_{Mea} + E_{Est_{t-1}}} \rightarrow E_{Est_t} = [1 - KG]E_{Est_{t-1}} \tag{4.7}$$

By using the previous filter, the velocity components of each detected object (4.4) are then solved optimally when a new bounding box is associated to an already detected bounding box. In the case of a new detection, the linear velocity model is used to make initial predictions. Those components are then used for the estimation of the future position of the bounding box. To accurately associate a bounding box to a detected pedestrian, the algorithm predicts its new location in the current frame.

To do that, it first computes the Jaccard distance (or Intersection over Union [IOU] distance) between the predictions and the detected objects by using the following method :

$$IOU = \frac{Area\,of\,Overlap}{Area\,of\,Union} \tag{4.8}$$

Choosing the IOU is a good way to reduce the impact of short-term occlusion on the tracking. After obtaining the corresponding matrix, the Hungarian algorithm is used in order to make the most cost effective choices in assigning detected objects to bounding boxes. This algorithm is used in order to resolve the data association problem (mentioned in section 2.1). In the context of this study, the two observation stages are two adjacent video frames. The parameter used for the comparison is the Jaccard Distance computed among detected objects in two video frames. The resulting matrix is used to tell the link between the observed components in an observation stage to another. This algorithm can be separated in two steps :

1. Creating the assignment matrices composed of the Jaccard distance between a previously identified bounding box and the newly detected bounding boxes
2. Mapping each previously detected bounding box by finding the optimal solution in the assignment matrix

The second step is done by applying the steps below to find the most optimal combination (Bruff, 2005):

1. Find the minimum value of each row [$rMin$]
2. Subtract $rMin$ to each row
3. Find the minimum value of each column [$cMin$]
4. Subtract $cMin$ to each column
5. Cover all the zeros in the matrix with the least amount of vertical and horizontal lines possible.
6. If the number of lines used is equal to the total number of lines then the optimal solution is found. If not, go to the next step.
7. Subtract the smallest uncovered value from all the non completely covered rows and add this value to all the completely covered columns. Finally go back to step 5.

Even so, a minimum IOU (IOU_{min}) is set in order to reject assignments where the overlap is less than the predefined value. By using the IOU_{min}, it is easy to identify untracked objects and assign a new id to entering objects. At first, the velocity is set to zero for newly detected objects. Each newly identified pedestrian passes a "probation" period where the computer makes sure they are not tracking an already identified pedestrian.

It is also significant to note that this algorithm has a short term memory in the sense that it does not support re-identification of objects, meaning it will count the same person twice if that pedestrian appears at two different times. This is due to the fact that tracked objects are terminated once they are undetected for a fixed number of frames (T_{lost}) and that it leads to more efficient use of computational resources.

4.2.2 Implementation

Using the merged detection_outputs.txt file created at the end of the detection step, we perform the SORT algorithm on this unified text file to extract the tracks. The input needs to strictly follow the format in equation 4.9 and can be modified using the parameters in table 4.3.

$$[x1, y1, x2, y2, score] \tag{4.9}$$

key	description
–max_age	Maximum number of frames to keep alive a track without an associated detection (T_{lost})
–min_hits	Minimum number of associated detections before a track is initialised
–iou_threshold	Minimum intersection over union threshold for a bounding box to be matched to one another. (IOU_{min})

Table 4.3: Input parameters for the tracking

As stated before, the first step revolves around the initialisation or update of the velocities of the detected objects using a Kalman Filter. In the code (*sort_detections.py* available on GitHub), it is done by using the Kalman Filter sub module of the filterpy library. It comes with an update function that assigns new values to unmatched detections and updates the values in already matched detections. The Kalman filter is also used to make predictions on the next state of the bounding boxes. For the data association step, a linear assignment function using the Hungarian algorithm is written to make the most optimal choice between the bounding box of a detected object and the predicted bounding boxes. The sorting script outputs a text file containing a bounding box with the following information, each line in this file associates a bounding box and its frame number with a track id :

$$[frame\ number, x1, y1, x2, y2, track\ id] \tag{4.10}$$

To visualise the tracks, we extract the center of each bounding box and then draw a line between each of those points based on the track id using the open cv library. We draw the tracks on the first frame of the video to better contextualise

where and how the tracks are distributed in the field of view (*draw_tracks.py* on GitHub).

4.3 Pedestrian data analysis

This section's objective is to explain the various methods used to extract information from the detections and tracks obtained from the previous sections. It is done in two parts : the first setting the theoretical basis of the methods used and the second explaining the various technical requirements needed to obtain the expected results.

Libraries used

All the scripts were written in Python or using the command prompt and use the following libraries :

- cv2
- numpy
- matplotlib : A library used to make static, dynamic and interactive figures in Python. For this study, it was mainly used in the analysis part to make adequate visualisations.

4.3.1 Theoretical basis

For the analysis of the results, this study will focus on extracting Origin/Destination counts based on the tracking explained in the precedent sections. This type of data is widely used in multiple scenarios such as in train stations (Hänseler, 2016), natural parks (Rupf et al., 2006) or on a city scale (Sevtsuk, 2021).

For the analysis and the creation of Origin/Destination counts we implemented a solution in Python using the line-line intersection algorithm (Cormen et al., 2022) to extract the entering and exiting tracks going over each counting line. This algorithm uses the slope of the line segments to make comparisons. The slope of a line segment of coordinates [(x1,y1),(x2,y2)] is calculated by using the following formula :

$$Slope = \frac{y2 - y1}{x2 - x1} \tag{4.11}$$

In order to know if a segment intersects another, the algorithm uses a concept called the orientation of an ordered triplet of points. Figure 4.7 showcases the three possibilities of orientation.

Based on those three orientations we can expand this concept to find if two segments intersect or not. Considering a segment A with a starting point $a1$ and an ending point $a2$ and a segment B with starting point $b1$ and ending point $b2$. Segments A and B intersect intersect only if one of the following cases is verified :

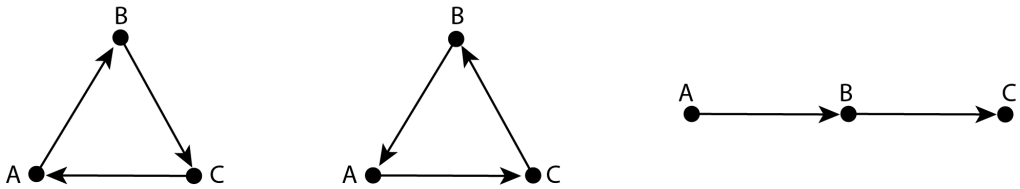


Figure 4.7: Clockwise, counterclockwise and collinear orientation of a triplet of points (from left to right) (Goodrich et al., 2014)

- General case :
  - $(a1, a2, b1)$ and $(a1, a2, b2)$ have different orientations **and**
  - $(b1, b2, a1)$ and $(b1, b2, a2)$ have different orientations
- Special case :
  - $(a1, a2, b1)$, $(a1, a2, b2)$, $(b1, b2, a1)$ and $(b1, b2, a2)$ are all collinear **and**
  - the x-projections of ($a1$,$a2$) and ($b1$,$b2$) intersect
  - the y-projections of ($a1$,$a2$) and ($b1$,$b2$) intersect

In order to compute the orientation of an ordered triplet of points [a,b,c], where $a1, b1, c1$ are the x-coordinates of the points and $a2, b2, c2$ are the y-coordinates. The following can be used:

$$\frac{b2 - a2}{b1 - a1} = \frac{c2 - b2}{c1 - b1} \tag{4.12}$$

This can also be written in the form of :

$$((b2 - a2) \times (c1 - b1)) - ((c2 - b2) \times (b1 - a1)) = value \tag{4.13}$$

This result can further be interpreted in the following ways :

- if $value > 0$ then the orientation is clockwise
- if $value < 0$ then the orientation is counterclockwise
- if $value = 0$ then the orientation is collinear

As the counted segments intersecting with the counting line are summed up, it is also possible to separate the origins and destinations by doing the cross product between the counting line and the intersecting segment. In order to compute this, the direction of the intersecting segment with the counting line must be extracted. In order to do so, it is necessary to compute the vector between the extremities of each segment. Considering the pair of points A and B, A having coordinates (x1,y1) and B having (x2,y2), equation 4.14 can be used to calculate the vector between those points.

$$\vec{AB} = (x_2 - x_1, y_2 - y_1) \tag{4.14}$$

Using the x and y coordinates of the points, the cross product of two dimensional vectors result in a scalar that can be interpreted in order to know the orientation of the intersecting segment. The formula to compute this is in equation 4.15.

$$Cross\ Product = x_1 \times y_2 - x_2 \times y_1 \tag{4.15}$$

The relation between the cross product and the the orientation between one another can be understood by the rule of thumb (Weisstein, 2024) and is easily interpretable since the vectors used in this study are two dimensional. This implies that the resulting value of our cross-product is a scalar that is positive in the case of an entering segment and negative in case of an exiting segment.

4.3.2 Implementation

For the methods described in the precedent section to work, it is necessary for the tracks and the counting lines to be set in a specific format. The tracks must be first aggregated into polyline objects in the format (4.17), where each object is composed of multiple pairs of coordinates for the extremities of each segment :

$$polyline = [[x_1, y_1], [x_2, y_2], ..., [x_n, y_n]] \tag{4.16}$$

$$polylines = [[polyline_1], [polyline_2], ..., [polyline_n]] \tag{4.17}$$

The counting lines are lines composed of 2 pairs of coordinates set in a clockwise manner (Fig. 4.8). Meaning the vector associated to the counting line will always be in the same perpendicular direction to the entryway. In this way, it is irrelevant at which angle or position in the image the counting line finds itself. This is done so that the result from the cross product can always be interpreted in the same way.

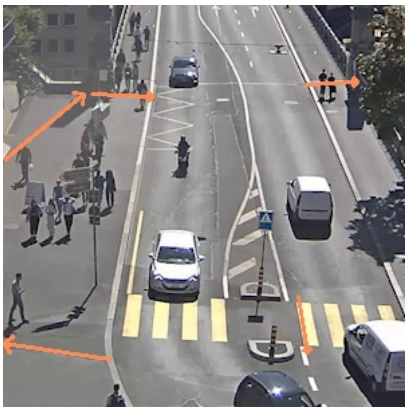


Figure 4.8: Orientation of the counting lines on video 01

The next step involves extracting the intersecting segments between the counting lines and the tracks. Following the general and special case identified in the last

section, the intersecting segments line segments are extracted (*count_OD.py* available on GitHub). Afterwards, calculating the cross-product between the intersecting segment and the counting line to identify the direction of the segment and know if the track is coming in or going out. This gives us the associated $track_{id}$ to the segment and therefore can let us sum up the number of unique tracks going in and out of each counting line.

Chapter 5

Results

In this chapter, the results of this study are discussed. Multiple detection algorithms and their associated results will be shown. In order to assess our results, a cross comparison will be done on two videos taken on the first of September 2023 and the tenth of October 2023, both preprocessed differently (see section 4.1.2 for details on the preprocessing steps), two differing detection thresholds and two tracking thresholds. This will be done by providing multiple images and tables relative to the amount and quality of the tracks and the detections. After exposing the results, an analysis will be done to explain some of the observed phenomena.

Table 5.1 contains the parameters used for the detection for this study and the acronyms associated with the results of the detections.

Threshold \ Model	MobileNet	PedNet
0.5	mnet0.5	pnet0.5
0.35	mnet0.35	pnet0.35

Table 5.1: Acronyms associated with the models and detection parameters

Table 5.2 shows the three combination of chosen parameters for the tracking. As recommended by our thesis supervisor, the three variants only vary based on the maximum amount of frames a track is kept alive without a new detection associated to it (T_{lost}). The videos having a 30 frames per second framerate, the T_{lost} parameter varies between one, two or three seconds.

Acronym	$30f$	$60f$	$90f$
Maximum age (frames per second)	30	60	90
Minimum hit	3		
IOU threshold	0.3		

Table 5.2: Tracking parameters and associated acronyms

The videos will also be named according to Table 5.3. The first frame of each video can be found in section A.3. These first frames will also be used to draw the identified tracks and be used as a canvas to visualise the affluence of pedestrians in each video.

As seen on the cropped images in the appendix (A.3), the change lies in the scale and the spot of the crop. The crop done on video S2 and O2 was done on a part of the image which, over the two videos, had more pedestrian traffic than the rest of

Video name	Description
S1	This video is the result of the first kind of preprocessing on the video taken on the first of September (A.2a).
S2	This video is the result of the second kind of preprocessing on the video taken on the first of September (A.2b).
O1	This video is the result of the first kind of preprocessing on the video taken on the seventeenth of October (A.2c).
O2	This video is the result of the second kind of preprocessing on the video taken on the seventeenth of October (A.2d).

Table 5.3: Videos used and the names associated to them

the image. This change in preprocessing was done in order to see if the difference in the size of the pedestrians (due to the change of scale) could have an impact on the results.

5.1 Detection, tracking and counting results

To show a general overview of the results of our study, table 5.4 shows the number of unique tracks identified on each video and with each combination of parameters according to the precedent naming conventions.

Detection parameters	SORT Parameters	S1	S2	O1	O2
mnet0.5	$30f$	7254	7875	5347	4468
	$60f$	7013	7603	5212	4332
	$90f$	6887	7454	5143	4264
mnet0.35	$30f$	12397	12342	8490	6817
	$60f$	11956	11839	8268	6592
	$90f$	11744	11596	8167	6496
pnet0.5	$30f$	5097	500	1147	73
	$60f$	4785	1032	459	70
	$90f$	4539	984	426	2
pnet0.35	$30f$	5760	575	1266	76
	$60f$	5416	512	1181	69
	$90f$	5156	464	1119	68

Table 5.4: Number of tracks extracted for each combination of parameters and videos

This lets us observe multiple general expected trends :

- Lower thresholds in detection lead to a higher number of tracks being identified.
- Increasing the maximum time a track can stay 'alive' without having a matched detection decreases the number of tracks. This decrease is pretty constant regardless of the video or other varying parameters.
- There is a similarity in the amount of detections between video S1/S2 and video O1/O2 using the MobileNet detection algorithm.

But also unexpected trends, such as :

- The PedNet algorithm has overall lower counts of tracked pedestrians than the MobileNet.
- The change in preprocessing shows an increase in the number of tracks identified for the mnet0.5 detections.
- The change in preprocessing drastically decreases the amount of tracks for the PedNet algorithm regardless of the detection threshold used.
- The MobileNet algorithm extracts more tracks than PedNet.

Another important part of our results lies in the extraction of Origin/Destination counts. The counting lines used for the extraction of the following tables can be found in the appendix (A.4) for each video. The numbering associated with each counting line is also on those images. As we can see in the resulting Origin/Destination counts (A.10), a majority of the tracks that are detected don't cross the counting lines or are not detected as such. We can also observe that the pretty constant variation that appeared between $30f$, $60f$ and $90f$ in table 5.4 is almost non existent. In order to fine tune our results, we will first analyze the results for each video individually and then proceed to the analysis.

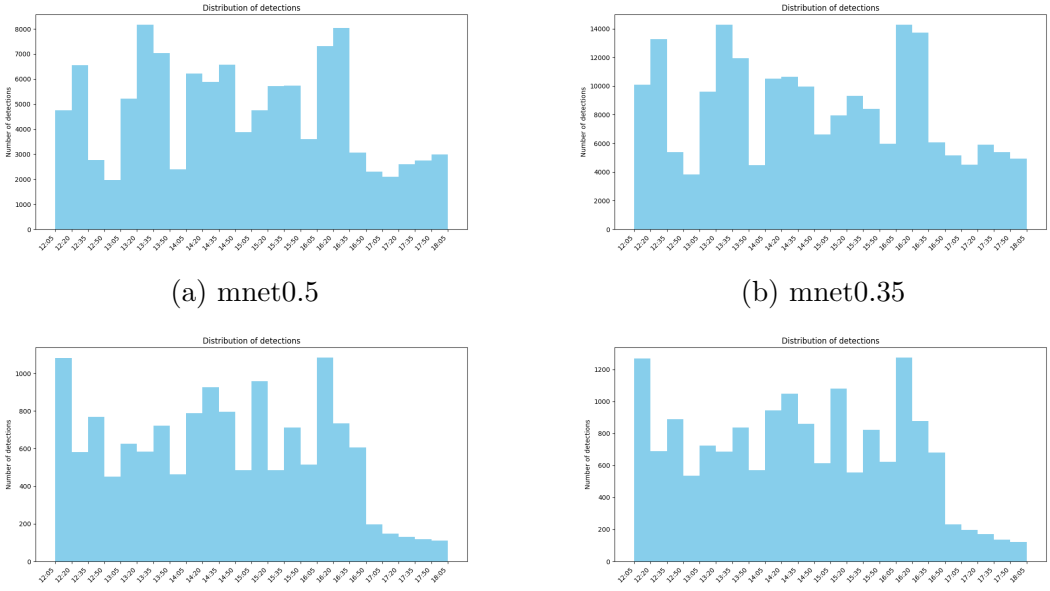
5.1.1 September 1st, first kind of preprocessing

Table 5.4 shows a large number of tracks across all combinations of parameters. As supposed before, we observe a constant decrease in the number of unique tracks with the change of the tracking parameters. Table 5.5 shows that for the same algorithm, there is a similar variation in the total of tracks. The change from $30f$ to $60f$ marks a bigger variation in the number of tracks. This can be due to the fact that keeping a track id alive for two seconds instead of one is a good way of increasing accuracy in such a study site. As this study site has multiple paths, there can be multiple spots where a pedestrian will enter into conflict with another moving object. Those zones of conflict can be the reason why a person will stop or change direction abruptly before resuming its itinerary so giving a 2 second leeway can make it possible for the tracking to account for those small changes.

	mnet0.5	mnet0.35	pnet0.5	pnet0.35
$30f \rightarrow 60f$	-241	-441	-312	-344
$60f \rightarrow 90f$	-126	-212	-246	-260
$30f \rightarrow 90f$	-367	-653	-558	-604
% change $30f \rightarrow 90f$	-5.06	-5.27	-10.95	-10.49

Table 5.5: Variation of tracks in video S1

Figures 5.1 show the distribution of the detections across the whole video. With each bin being an interval of fifteen minutes, the video is subdivided in twenty four time periods.



(a) mnet0.5 (b) mnet0.35

(c) pnet0.5 (d) pnet0.35

Figure 5.1: Distribution of detections in video S1

Those histograms let us see that our models identify regular variations in the number of detections. A hypothesis for the variation of detections is due to the hourly end of class time happening at the nearby gymnasium. This can also explain why there is such an almost hourly rising increase in the detections for this video. At around five in the afternoon, a decline in the number of detections in all the histograms can be seen. For the MobileNet, this decrease can be explained by the fact that the last students of the nearby high school are gone at this hour on a Friday evening. But the decrease in the PedNet detections are far sharper than the MobileNet ones. A hypothesis to explain this difference is the extension of the shadow across half of the field of view which considerably impacts the luminosity of the image (as pictured in image 5.2). The combination of both the end of the day and the end of school period could be the explaining factors for such changes. MobileNet seems to be less impacted by the change in luminosity due to the fact that the number of detections still matches the down times captured during the afternoon.

The figures at appendix A.6 lets us get a overview of the drawn tracks for each scenario. This confirms that the reduction of the detection threshold effectively sharply increases the number of tracks being identified. The most notable observation from those images is related to the difficulty of both algorithms in completing the task over the pedestrian crossing. For MobileNet, some tracks can be seen but there seems to be a blank spot in the middle aisle where our algorithms have some difficulty in following the pedestrians. This could be related to the sudden change in background colour, the presence of urban infrastructure or in some cases the change in velocity when crossing the street. The PedNet algorithm shows a complete lack of tracks over the pedestrian crossing. Based on the previous hypothesis of a higher sensitivity to light changes, the zebra markings can be the reason why the PedNet

Figure 5.2: Video S1 at 4:50pm

has not detected any tracks over this part of the video.

The O/D counts are in appendix A.10. Most of the tracks don't cross the counting lines and therefore this leads to such results in the counting of origin/destinations for each counting line. This could be an issue with model training or depth perception. One solution to enhance the counts and guarantee a good counting result is to move the counting lines away from the border of the image. This is done in response to the minimum hit parameter of the SORT algorithm which requires for an object to be detected thrice before it can be made into a track. Alternatively, lowering this parameter could improve the results at the risk of adding even more noise to the results at the cost of accuracy. The O/D counts don't show as much variation between $30f$, $60f$ and $90f$ as the absolute number of tracks. That could be due to the fact that most of the counting lines are passing points where pedestrians will not suffer from too much variation in velocity and direction. This leads to a better quality in tracking using the Kalman Filter.

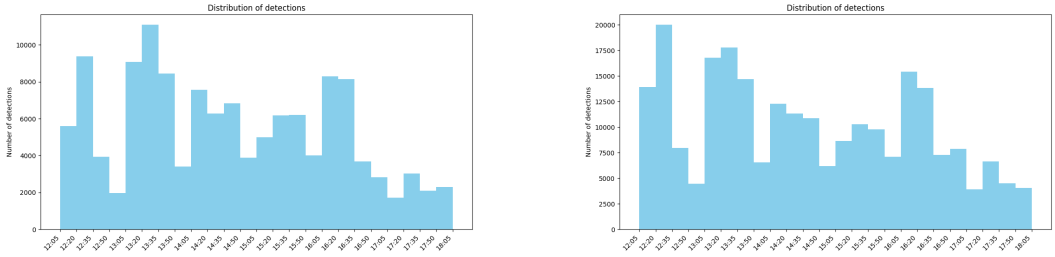
5.1.2 September 1st, second kind of preprocessing

As stated in the general trends, the MobileNet results for video S2 are very similar to those of video S1. That is not the case for PedNet where we see a very strong decrease in the amount of tracks between video S1 and video S2.

	mnet0.5	mnet0.35	pnet0.5	pnet0.35
$30f \rightarrow 60f$	-272	-503	-41	-63
$60f \rightarrow 90f$	-149	-243	-33	-48
$30f \rightarrow 90f$	-421	-746	-74	-111
% change $30f \rightarrow 90f$	-5.34	-6.04	-14.8	-19.3

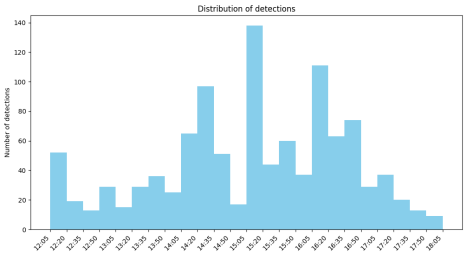
Table 5.6: Variation of tracks in video S2

The variation of tracks with the tracking parameters stays consistent with the precedent results, with a smaller variability for the MobileNet results and a higher one for PedNet. This further confirms that the impact of putting 2 seconds of T_{lost} has a more significant impact on the tracking process than putting 3 does.

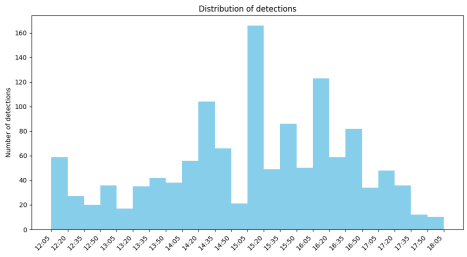


(a) mnet0.5

(b) mnet0.35



(c) pnet0.5



(d) pnet0.35

Figure 5.3: Distribution of detections in video S2

The same hourly variation can be observed on the mnet0.5 and mnet0.35 histograms as for video S1. This shows that the difference in preprocessing has little impact on the MobileNet results. By analysing the images at appendix A.7, we can observe that the algorithm does a better job at tracking pedestrians across the crossing. This is also most likely due to the difference in preprocessing. In this video, pedestrians are larger and can therefore be recognised with higher confidence compared to video S1 and O1.

The PedNet model is very heavily influenced by the change in preprocessing so it is not possible to extract the same patterns that were found in the first video. One thing to note is a sudden spike in the detections in the middle of the detection period (around 3pm) with no particular event or massive group of pedestrians going through the field of view. Appendix A.7 displays the minimum difference in the amount of tracks between $30f$, $60f$ and $90f$ for MobileNet. As opposed to video S1, there is more tracks being drawn directly on the cars path. This potentially means that MobileNet does a worse job at differentiating cars and pedestrians at this scale or that it can recognise the persons sitting in cars and therefore classify them as pedestrians. The PedNet model has almost no results to show apart from some specks. We can still see a complete absence of tracks on the zebra crossing.

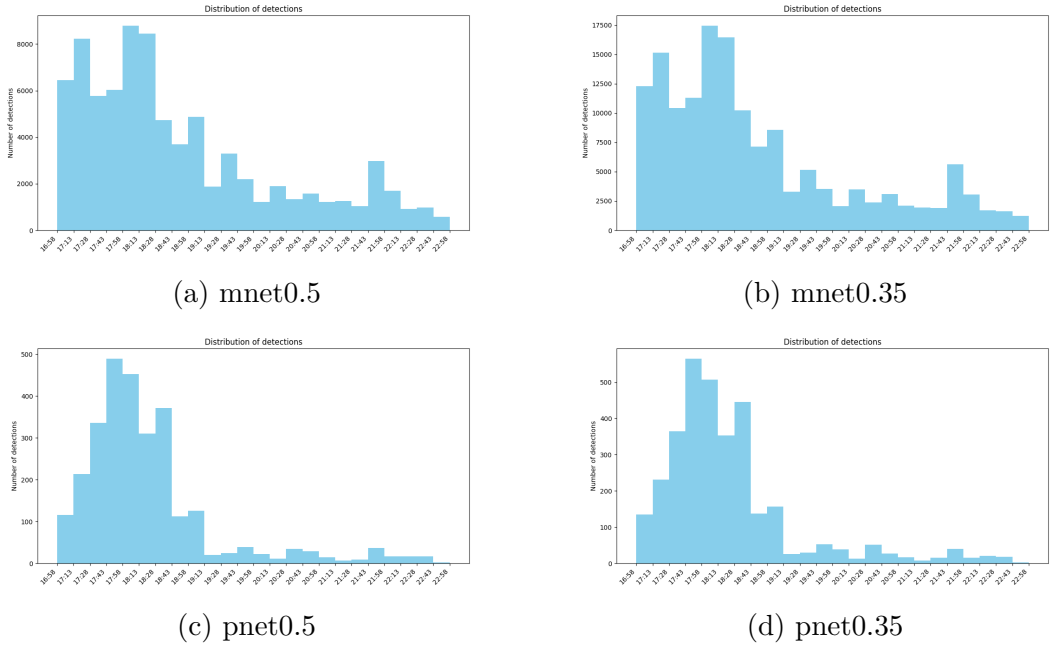
The O/D counts for this video are available in the appendix (A.11). For MobileNet, we can see that counting line 2 and 4 are well tracked and show some results. The other pair have less counts than expected. The zebra crossing counting line does not give any information even though there are drawn tracks on it. This is probably due to the position of the counting line being too close to the corner of the image.

5.1.3 October 17th, first kind of preprocessing

In Table 5.4, we can observe a large disparity between the two algorithms. The unique tracks identified by the MobileNet are far more numerous than its PedNet counterpart. The decrease in unique tracks related to the change in tracking parameters is very different depending on the algorithm. Table 5.7 shows that, in the case of the mnet0.5 and mnet0.35, there is a -3.81% and -3.8% variation, we can infer that the model does a good job at tracking with the initial parameters for this video. The pnet0.5 and pnet0.35 show a much higher variation (-14.21% and -11.61%).

	mnet0.5	mnet0.35	pnet0.5	pnet0.35
$30f \rightarrow 60f$	-135	-222	-115	-85
$60f \rightarrow 90f$	-69	-101	-48	-62
$30f \rightarrow 90f$	-204	-323	-163	-147
% change $30f \rightarrow 90f$	-3.81	-3.8	-14.21	-11.61

Table 5.7: Variation of tracks in video O1



(a) mnet0.5

(b) mnet0.35

(c) pnet0.5

(d) pnet0.35

Figure 5.4: Distribution of detections in video O1

Figure 5.4 shows a clear drop in the number of detections around the 7pm mark. This observation can be done across all detection parameters. This drop can be due to multiple factors such as the change in lighting due to the sun fall and the activation of street lighting. But another possible cause would be the end of the work or school day and a lower amount of traffic. Figure 5.5 shows the contrast of lighting before 7pm and after 7pm. Those histograms also show that there is a very high variability in the number of detections. The detection threshold plays a very significant role for the MobileNet and the PedNet. But even if the number of detections is a lot higher the lower the threshold is, the patterns identified by the algorithms are almost identical. The MobileNet identifies two peaks and the PedNet identify a single one.

(a) Snapshot of video O1 at 6:30pm

(b) Snapshot of video O1 at 7:30pm

Figure 5.5: Lighting contrast in video O1

The images at A.8 confirm the above mentioned discrepancy between the two algorithms. The MobileNet does a relatively good job at tracking objects of interest across the whole image but the PedNet seems to not be able to track pedestrians on the majority of the image but the right side of the bridge. Also there is still not even a trace of a track on the pedestrian crossing as seen in video S1. This further confirms the hypothesis that the PedNet either has trouble detecting pedestrians over the zebra crossing or that the bounding boxes resulting from detections over the crossing are so different from the ones before that the data association step of our algorithm can not be successfully completed. The images resulting from the MobileNet show the same hole in the middle of the pedestrian crossing that was observed in video S1. One particular thing about the tracks is the presence of a lot of them on the road on figure A.12 and A.13. This is probably due to some cars being detected as pedestrians.

The Origin Destination counts associated to video O1 are at appendix A.12. As expected from the aforementioned observations, there is a stark contrast in the number of counts done on the PedNet and MobileNet ones. In the case of MobileNet, we can see that there is two counting lines that have performed a lot better than the rest. Number 3 and 5 are the counting lines that are on the right side of the bridge and in the bottom left corner respectively.

5.1.4 October 17th, second type of preprocessing

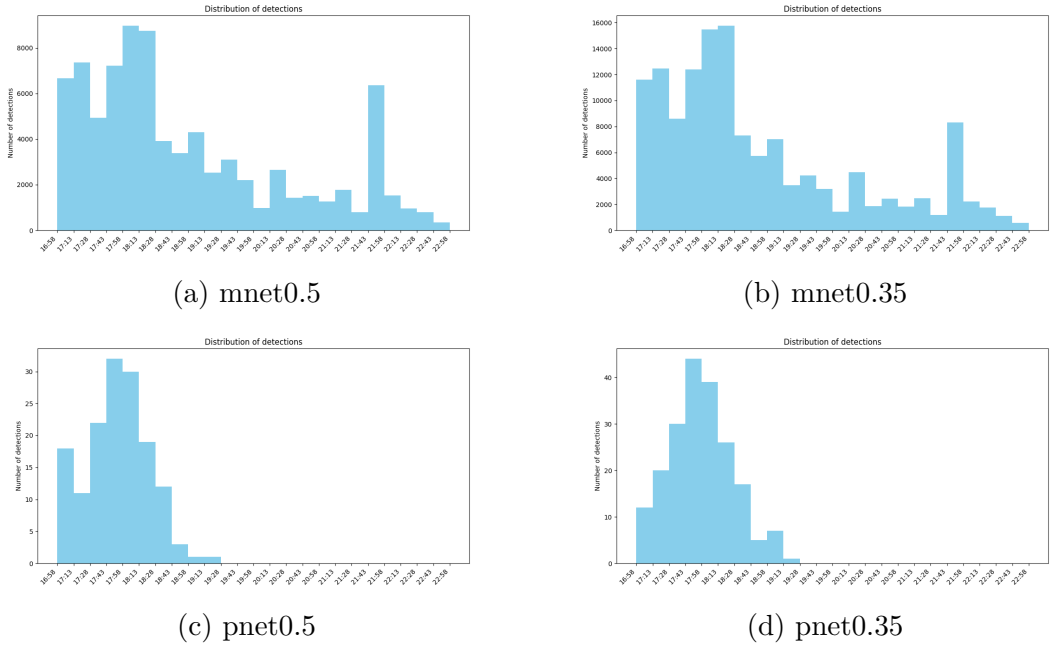
Video O2 is the video that has by far the least amount of unique tracks across all detection models. This may partly be due to a smaller range of detection due to the change of scale or, in the case of PedNet, a combination of aggravating factors such as the change in lighting (as pictured in figure 5.5) or the preprocessing difference resulting in a worse performance.

As seen in Table 5.8, the variation of unique tracks in mnet0.5, mnet 0.35 and pnet0.35 are very similar to the ones observed on the other videos but not for pnet

	mnet0.5	mnet0.35	pnet0.5	pnet0.35
$30f \rightarrow 60f$	-136	-225	-3	-7
$60f \rightarrow 90f$	-68	-96	-68	-1
$30f \rightarrow 90f$	-204	-321	-71	-8
% change $30f \rightarrow 90f$	-4.56	-4.7	-97.26	-10.52

Table 5.8: Variation of tracks in video O2

0.5, where the decrease is very important (-97.26%).



(a) mnet0.5

(b) mnet0.35

(c) pnet0.5

(d) pnet0.35

Figure 5.6: Distribution of detections in video O2

Figures 5.6 show us the same patterns for the MobileNet where we see a gradually decreasing number of detection until 7:30pm where they stabilise with some occasional peaks. The PedNet model did a very bad job at detecting pedestrians as it stopped detecting pedestrians at approximately 7:30pm too. This can be seen by the absence of detections from that point on.

Figures 5.6a and 5.6b further confirm the need for the counting lines to be positioned a bit further from the border in order to guarantee a good count. It is also possible to see a lot of noise in the mnet0.35 version of the results.

The PedNet results are completely empty and show no tracks apart from a single one on the top right of the image (5.6c) that is most likely a car due to its position. Those empty results are therefore translated into completely empty origin destination counts. This shows that the combination of dimming lights and this configuration in terms of scale is not appropriate for the PedNet.

Chapter 6

Discussion

In this chapter, we will first discuss the results of the previous chapter and delve deeper into the analysis. Following this, the limitations that were encountered while realising this study will be discussed. Also future works and some potential improvements will be described.

6.1 Analysis

As we can see from the precedent observations, there is a vast disparity between the detection models. The MobileNet performs far better than the PedNet one. This can be seen by the amount of tracks and the visualisations for each scenario. The MobileNet seems to be more resilient to change and adapts better to variability in external factors such as shadow contrast, or illumination variation. The drop in the amount of detections that happens with the appearance of shadows on video S1/S2 for pnet0.5 and pnet0.35 is a clear indication of that. Also the overall difficulty of all models in detecting and tracking accurately once the night falls further proves the sensibility of such methods to environmental variations. To reduce this vulnerability, choosing a detection algorithm that was trained on a wider range of images like MobileNet is a better way to make assumptions regarding the pedestrian patterns and to extract useful data from such video feeds. Even then, it is crucial to combine the insights extracted from the detections histogram, the visualisation of the tracks and the count of origin destination for each counting line. This estimation can be used to make accurate planning decisions.

In order to minimise the computational cost of the tracking while having the best results, the best configuration of parameters would be to set the T_{lost} to 60 frames (2 seconds). This is also so that the Kalman filter used by the SORT algorithm can be used in the most efficient way to predict the next state of the detected bounding box, as it can not always take into account minute direction or velocity variations. Of course, in the case of smaller datasets, the difference will not be as obvious. If the number of detections gets larger then the tracking process gets more time-consuming, so the trade off between execution time and quality results need to be taken into account. The number of detections and the resulting tracks vary with the chosen detection threshold as seen in the figures A.5, A.6, A.8, A.7 and A.9. This led us to believe that choosing a threshold between 0.3 and 0.5 would provide the

best results while removing the most noise and as such creating less errors in the detections.

Although mnet0.5 and mnet0.35 have less trouble detecting in a wider range of conditions, we can observe the rise in the detection of noise or other objects as pedestrians. Notably in figure A.9 where a lot noise can be seen on the road. So although lowering the detection threshold increases the number of detections, the results are more prone to errors in detection. One important defect in the findings of this study is the difficulty in tracking pedestrians across the zebra crossing which represented a main axis in the traffic flow of the study site.

All those findings highlight a central component of our pipeline which is the choice of the detection model which directly impacts on the overall quality of our results and the ability of our pipeline to make accurate observations and extract accurate and relevant insights. This also lets us confidently say that using the proposed pipeline works and has led us to acquire a more thorough understanding of the optimal methods and parameters used for pedestrian data extraction and analysis.

6.2 Limits

This study was done to construct a detection and tracking pipeline to be used on portable devices and with the objective of tracking and counting pedestrians. We therefore exclude the other modes of transport such as cars, bicycles and public transport. Also, this study was performed in an urban environment where there is a clear organisation of pedestrian pathways which is not the case in every scenario. The whole study uses a particular set of hardware and it was not tried on an another set of cameras, this could lead to variations in the results.

The sample of videos used for this study was kept on the lower end, for us to perform a wide range parameter variation testing. Those parameters were :

- The preprocessing parameters such as scale and position that are associated with the cropping of the videos to an accepted format by the detection algorithm.
- The detection confidence threshold used to control which detections were written to the output file based on the confidence score associated with them
- The tracking parameter T_{lost} which is the number of frames during which a track is kept 'alive' without an associated detection. This is a parameter that is increased in order to reduce the impact of small direction or illumination variation and occlusion.

This lower number of initial videos may hinder the generalisation of the observed optimal parameters and the performance of the algorithms in varying scenarios.

The choice of the detection algorithm was constrained by the ones available on the initial jetson-inference GitHub repository. This is partly due to the constraint of using the Jetson Nano Developer kit for the detection part of our pipeline, as such we chose ready-to-use models on this type of device. Using those models and the architecture associated with them proved to be the right choice as the compatibility between the Jetson and the repository was perfect. The initial counting project that we took inspiration from used the SORT algorithm and did the detection on a Jetson too. The results were very similar to our expected results. As such we chose the SORT algorithm even though it is not the latest model for tracking. It is one on which a lot of development has been done such as DeepSORT or Bot-SORT.

Using other video sources could have further diversified our results and proven some of our observations relative to the sensitivity of the models to illumination and preprocessing variations. The impact of those variations can be seen by the fact that all the models experienced a significant drop in the amount of detections once the lights dimmed and the source of illumination changed to street lights. Although, since the models were pre-trained, the results of this study also give us insights into the adaptability of the models in front of new data. Making sure the MobileNet adapts to various scenarios could prove to be crucial in order to extract valuable information from its detections.

The findings of this study can be generalised to similar kind of videos but further testing would need to be done in order to prove if a change in hardware would have an impact on the detections and the subsequent tracking process.

It is important to note that the data used for this study is publicly available and can be used as stated on the official website of Lausanne about its use of video surveillance [2]. This webcam is installed because of its wide angle and the impossibility to recognise particular features on specific pedestrians. Therefore it is in accordance with data privacy laws and can safely be used. In other scenarios such as the Lavaux one, there was a need for conversation with the local authorities in order to install camera devices on public properties.

6.3 Improvements

The development of an effective pipeline for extracting reliable data from video cameras and then analysing this data is still a subsequent effort and is faced with multiple difficulties as seen in the last section. Overall the emergence of a multitude of open source tools that can be programmed to answer the current societal problems will slowly lead to even better performances. But as of now, there still exist some challenges to be addressed to provide reliable data and a simple way of acquiring it.

The first main improvement can be done on the choice of algorithms for the

[2]FAQ related to video surveillance for the city of Lausanne: https://www.lausanne.ch/officiel/administration/securite-et-economie/police-de-lausanne/a-propos/videosurveillance/faq-videosurveillance.html

detection and tracking part of the study. Using state of the art methods and implementing those, it is expected that there would be a substantial increase in accuracy. Recently, works on deep-learning techniques for detection and tracking have been researched and could be used to fulfill the same goals as those of this study. They are still not as lightweight as needed for a portable system such as the one running on the Jetson Nano Developer but as the technology matures, there will be a reduction in the computational cost. Or it could come from the development of more compact and powerful hardware that is capable of running such algorithms for longer times.

Apart from the choice of algorithm, the main way of increasing the accuracy of our results, regardless of the method used, is to train our model on the dataset to be used. But this solution comes with an additional need for resources in labelling and creating data. But the trade off can be good if the spot is to be repeatedly used or at least in similar locations where the trained model could have a good enough performance with some calibration. As we see in this study, choosing the most optimal combination of parameters can lead to vast differences. So preprocessing the videos in a way that the objects of interest look almost the same as the objects on which the model was trained on could lead to far better performances for the detection, tracking and counting.

Alternative ways of counting and extracting O/D counts using ReID and transfer learning are being developed (Wong et al., 2022) to extract pedestrian data on bigger scales and using multiple cameras in different spots.

More related to the results of this study, using a wider range of videos could further confirm or deny the claims that were made in the preceding sections.

One important point to note is that using multiple sensors to detect the same phenomenon can often lead to a rise in the accuracy of the results due to the cross-comparison of results and by using each sensors to their advantage. There is no single sensing technique that fits all purposes. Instead, often the simultaneous consideration of multiple data sources yields the most reliable results. Just as stated by Labbe (2020) in his work on Kalman filters : «[...] two sensors, even if one is less accurate than the other, is better than one».

Chapter 7

Conclusion

In pursuit of creating a simple pipeline for the analysis of pedestrian data using camera sensors, this study has delved into the extraction of reliable data from raw video inputs and the analysis of this data. Firstly, we can can conclude that :

- There exists a large variability in the results extracted depending on the detection model.
- The training of a model has a large impact on the results too as the more similar a training dataset is to the new data, the better the results will be.
- The parameters of each method are central in obtaining adequate results. This is true at every step of the pipeline.
- Environmental factors play a very important role in the quality of the results.
- If the input images are preprocessed, it also plays a pivoting role in the quality of the extracted data.

In comparing the extractions of the PedNet and MobileNet algorithms, this study provides valuable insights into the capacity of those models to make accurate detections in various parameter combinations. Also while the change in parameters for the SORT algorithm gave more similar results across all combinations bar a few exceptions, it did however confirm the need to adapt some parameters according to the study site to account for site specific behaviours due to congestion, interaction with other modes of transport and change in pathway width.

Using the findings from the first part of the pipeline, it was possible to extract insights in regard to the use of urban spaces and pedestrian behaviour patterns :

- As presupposed, it was possible to identify the rush hours and the spatial distribution of the pedestrians.
- Using the O/D counts it was also possible to identify the main foot ways.

While those findings provide valuable insights in the topic of counting pedestrians from camera sources, it is important to keep in mind that there are limitations related to the use of cameras for such purposes and that using other tools such as Bluetooth, GPS or mobile data in combination with camera sensors for validation

will definitely provide higher accuracy results than focusing on a single type of sensor.

In conclusion, this study integrates itself in the work of sustainable cities in promoting walking as a mode of transport and can be used as a starting point to provide insights into how spaces are being used. This work shows the feasibility of setting up such pedestrian counting systems from video sources in urban spaces. Even so, it does need to go through a parameter search particular to each study site based on various factors. Using those, open source tools can be used to reproduce the pipeline and aid in further urban development.

References

actif-trafiC. (2022, November 10). *Marchabilité et santé – comparaison entre les villes suisses*. Retrieved January 18, 2024, from https://www.actif-trafic.ch/marchabilite

Barandiaran, J., Murguia, B., & Boto, F. (2008). Real-time people counting using multiple lines. *2008 Ninth International Workshop on Image Analysis for Multimedia Interactive Services*, 159–162.

Baviera-Puig, A., Buitrago-Vera, J., & Escriba-Perez, C. (2016). Geomarketing models in supermarket location strategies. *Journal of Business Economics and Management*, *17*(6), 1205–1221. https://doi.org/10.3846/16111699.2015.1113198

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*, 3464–3468. https://doi.org/10.1109/ICIP.2016.7533003

Black, J., Ellis, T., & Rosin, P. (2002). Multi view image surveillance and tracking. *Workshop on Motion and Video Computing, 2002. Proceedings.*, 169–174. https://doi.org/10.1109/MOTION.2002.1182230

Bruff, D. (2005). *The assignment problem and the hungarian method*. Retrieved January 5, 2024, from https://math.harvard.edu/archive/20_spring_05/

CARTO. (2024, January 17). *Harness spatial analysis for geomarketing — CARTO*. Retrieved January 17, 2024, from https://carto.com/solutions/geomarketing

Cessford, G., & Muhar, A. (2003). Monitoring options for visitor numbers in national parks and natural areas. *Journal for Nature Conservation*, *11*(4), 240–250. https://doi.org/10.1078/1617-1381-00055

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms, fourth edition*. MIT Press.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, *1*, 886–893 vol. 1. https://doi.org/10.1109/CVPR.2005.177

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1627–1645. https://doi.org/10.1109/TPAMI.2009.167

Gehl, J. (2013). *Cities for people*. Island Press.

Géron, A. (2022). *Hands-on machine learning with scikit-learn, keras, and TensorFlow*. "O'Reilly Media, Inc.".

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation, 580–587. Retrieved December 26, 2023, from https://openaccess.thecvf.com/content_cvpr_2014/html/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.html

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014, January 28). *Data structures and algorithms in java*. John Wiley & Sons.

Han, T., Bai, L., Gao, J., Wang, Q., & Ouyang, W. (2022). DR.VIC: Decomposition and reasoning for video individual counting, 3083–3092. Retrieved June 14, 2023, from https://openaccess.thecvf.com/content/CVPR2022/html/Han_DR.VIC_Decomposition_and_Reasoning_for_Video_Individual_Counting_CVPR_2022_paper.html

Hänseler, F. S. (2016). *Modeling and estimation of pedestrian flows in train stations* (Doctoral dissertation). EPFL. Lausanne. https://doi.org/10.5075/epfl-thesis-6876

Hollenhorst, S. J. (1992). *Monitoring visitor use in backcountry and wilderness: A review of methods*. Pacific Southwest Research Station.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. https://doi.org/10.48550/arXiv.1704.04861

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35–45. https://doi.org/10.1115/1.3662552

Labbe, R. (2020, May 23). *Kalman and bayesian filters in python*.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015). Microsoft COCO: Common objects in context. https://doi.org/10.48550/arXiv.1405.0312

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot MultiBox detector. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision – ECCV 2016* (pp. 21–37). Springer International Publishing. https://doi.org/10.1007/978-3-319-46448-0_2

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

Martin, S. (2020, October 23). *What is computer vision?* [NVIDIA blog]. Retrieved December 27, 2023, from https://blogs.nvidia.com/blog/what-is-computer-vision/

Michel van Biezen. (2015, September 13). *Special topics - the kalman filter (2 of 55) flowchart of a simple example (single measured value)*. Retrieved January 18, 2024, from https://www.youtube.com/watch?v=tk3OJjKTDnQ

Montgomery, M. R., Stren, R., Cohen, B., & Reed, H. E. (2013, October 31). *Cities transformed: Demographic change and its implications in the developing world*. Routledge.

Nieuwenhuijsen, M. J. (2020). Urban and transport planning pathways to carbon neutral, liveable and healthy cities; a review of the current evidence. *Environment International*, *140*, 105661. https://doi.org/10.1016/j.envint.2020.105661

NVIDIA. (2023, December 31). *Jetson nano developer kit* [NVIDIA developer]. Retrieved December 31, 2023, from https://developer.nvidia.com/embedded/jetson-nano-developer-kit

Papageorgiou, C., & Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision*, *38*(1), 15–33. https://doi.org/10.1023/A:1008162616689

Papers with code - object detection. (2023, December 26). Retrieved December 26, 2023, from https://paperswithcode.com/task/object-detection

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection, 779–788. Retrieved November 15, 2023, from https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html

Ren, W., Wang, X., Tian, J., Tang, Y., & Chan, A. B. (2021). Tracking-by-counting: Using network flows on crowd density maps for tracking multiple targets. *IEEE Transactions on Image Processing*, *30*, 1439–1452. https://doi.org/10.1109/TIP.2020.3044219

Ribeiro, F. R., Silva, A., Barbosa, F., Silva, A. P., & Metrôlho, J. C. (2018). Mobile applications for accessible tourism: Overview, challenges and a proposed platform. *Information Technology & Tourism*, *19*(1), 29–59. https://doi.org/10.1007/s40558-018-0110-2

Rupf, R., Wernli, M., & Filli, F. (2006). Visitor counting with acoustic slab sensors in the swiss national park. *Exploring the Nature of Management*, 72–77.

Rupf, R., Wernli, M., & Haller, R. (2008). How to elaborate precise visitor numbers. *Management for protection and sustainable development. Proceedings of the Fourth International Conference on Monitoring and Management of Visitor Flows in Recreational and Protected Areas. Montecatini Terme, Italy*, 161–164. Retrieved October 4, 2023, from https://www.academia.edu/download/76676235/rupf_reto_wernli_-2008-how_to_elaborate_pre.pdf

SBB. (2023, April 10). *Pas de reconnaissance faciale dans les gares* [SBB News]. Retrieved October 4, 2023, from https://news.sbb.ch/fr/article/115997/pas-de-reconnaissance-faciale-dans-les-gares

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2014, February 23). OverFeat: Integrated recognition, localization and detection using convolutional networks. https://doi.org/10.48550/arXiv.1312.6229

Sevtsuk, A. (2021). Estimating pedestrian flows on street networks [Publisher: Routledge _eprint: https://doi.org/10.1080/01944363.2020.1864758]. *Journal of the American Planning Association*, *87*(4), 512–526. https://doi.org/10.1080/01944363.2020.1864758

Shinozuka, M., & Mansouri, B. (2009, January 1). 4 - synthetic aperture radar and remote sensing technologies for structural health monitoring of civil infrastructure systems. In V. M. Karbhari & F. Ansari (Eds.), *Structural health monitoring of civil infrastructure systems* (pp. 113–151). Woodhead Publishing. https://doi.org/10.1533/9781845696825.1.114

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going deeper with convolutions. https://doi.org/10.48550/arXiv.1409.4842

Tao, A., Barker, J., & Sarathy, S. (2016, August 11). *DetectNet: Deep neural network for object detection in DIGITS*. Retrieved January 18, 2024, from https://developer.nvidia.com/blog/detectnet-deep-neural-network-object-detection-digits/

United Nations. (2022). *World population prospects 2022: Summary of results*. https://doi.org/10.18356/9789210014380

United Nations. (2023). Goal 11: Sustainable cities and communities — sustainable development goals — united nations development programme. Retrieved October 4, 2023, from https://www.undp.org/sustainable-development-goals/sustainable-cities-and-communities

van der Spek, S. (2009). Mapping pedestrian movement: Using tracking technologies in koblenz. In G. Gartner & K. Rehrl (Eds.), *Location based services and TeleCartography II: From sensor fusion to context models* (pp. 95–118). Springer. https://doi.org/10.1007/978-3-540-87393-8_7

Ville de Lausanne. (2020, November 6). *Lausanne, pont bessières*. Retrieved January 19, 2024, from https://www.youtube.com/watch?v=y3sMI1HtZfE

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, *1*, I–I. https://doi.org/10.1109/CVPR.2001.990517

Weisstein, E. W. (2024, January 18). *Cross product*. Retrieved January 18, 2024, from https://mathworld.wolfram.com/

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. *2017 IEEE International Conference on Image Processing (ICIP)*, 3645–3649. https://doi.org/10.1109/ICIP.2017.8296962

Wong, P. K.-Y., Luo, H., Wang, M., & Cheng, J. C. P. (2022). Enriched and discriminative convolutional neural network features for pedestrian re-identification and trajectory modeling. *Computer-Aided Civil and Infrastructure Engineering*, *37*(5), 573–592. https://doi.org/10.1111/mice.12750

Yoshinaga, S., Shimada, A., & Taniguchi, R.-i. (2009). Real-time people counting using blob descriptor. *Procedia - Social and Behavioral Sciences*, *2*(1), 143–152. https://doi.org/10.1016/j.sbspro.2010.01.028

Appendix A

A.1 MobileNet Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 128$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Table A.1: Architecture of MobileNet model taking a $224 \times 224 \times 3$ input image (Howard et al., 2017)

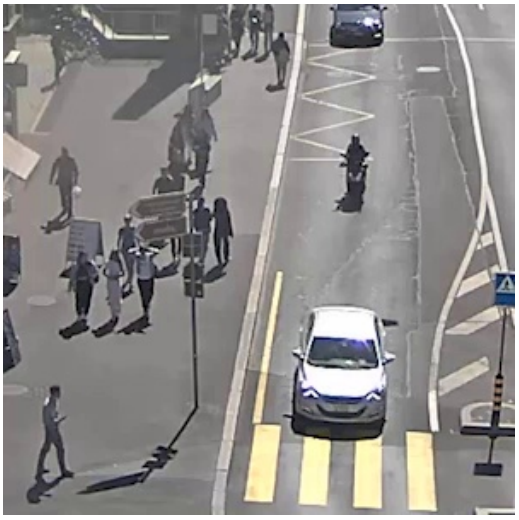
A.2 COCO labels

class name	class id	class name	class id
unlabeled	0	wine glass	46
person	1	cup	47
bicycle	2	fork	48
car	3	knife	49
motorcycle	4	spoon	50
airplane	5	bowl	51
bus	6	banana	52
train	7	apple	53
truck	8	sandwich	54
boat	9	orange	55
traffic light	10	broccoli	56
fire hydrant	11	carrot	57
street sign	12	hot dog	58
stop sign	13	pizza	59
parking meter	14	donut	60
bench	15	cake	61
bird	16	chair	62
cat	17	couch	63
dog	18	potted plant	64
horse	19	bed	65
sheep	20	mirror	66
cow	21	dining table	67
elephant	22	window	68
bear	23	desk	69
zebra	24	toilet	70
giraffe	25	door	71
hat	26	tv	72
backpack	27	laptop	73
umbrella	28	mouse	74
shoe	29	remote	75
eye glasses	30	keyboard	76
handbag	31	cell phone	77
tie	32	microwave	78
suitcase	33	oven	79
frisbee	34	toaster	80
skis	35	sink	81
snowboard	36	refrigerator	82
sports ball	37	blender	83
kite	38	book	84
baseball bat	39	clock	85
baseball glove	40	vase	86
skateboard	41	scissors	87
surfboard	42	teddy bear	88
tennis racket	43	hair drier	89
bottle	44	toothbrush	90
plate	45		

Table A.2: COCO labels (Lin et al., 2015)

A.3 Cropped images

(a) 01.09.2023 - First frame of video S1

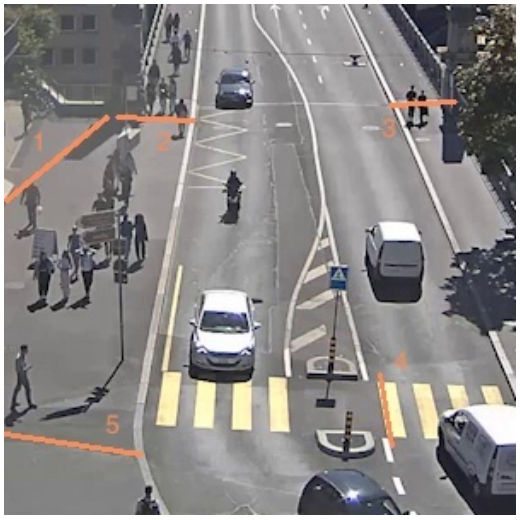


(b) 01.09.2023 - First frame of video S2

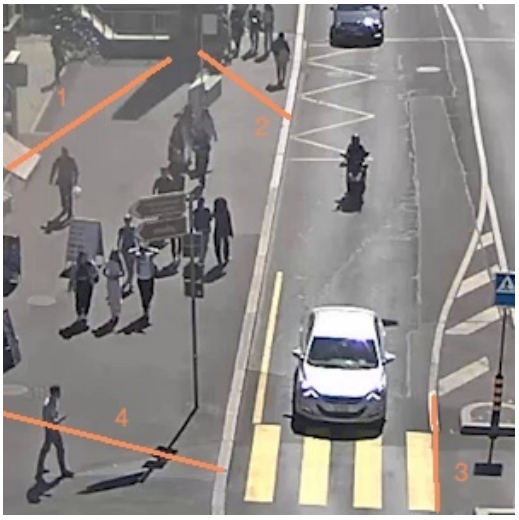
(c) 17.10.2023 - First frame of video O1

(d) 17.10.2023 - First frame of video O2

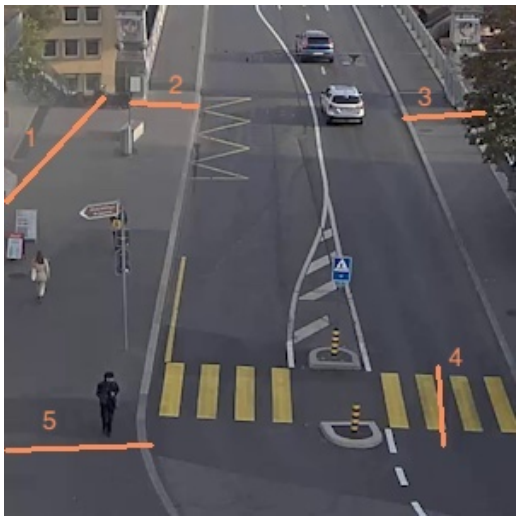
A.4 Counting Lines



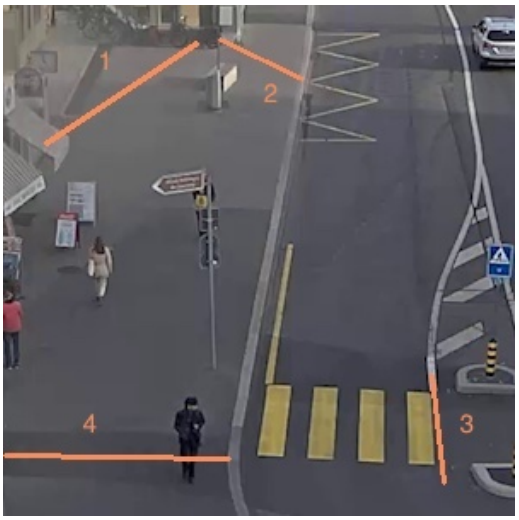
(a) 01.09.2023 - Counting lines on video S1



(b) 01.09.2023 - Counting lines on video S2



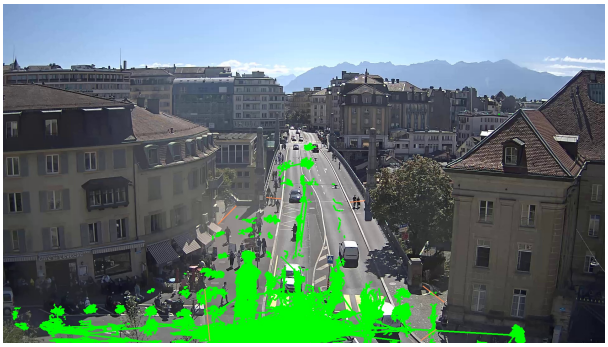
(c) 17.10.2023 - Counting lines on video O1



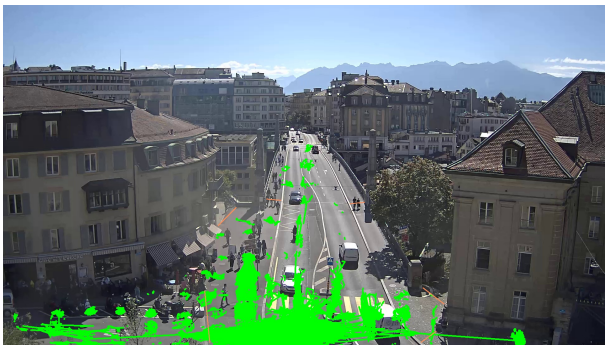
(d) 17.10.2023 - Counting lines on video O2

A.5 Variation in minimum detection threshold

This appendix contains the visual results on the video of Bessieres bridge of the first of September. The algorithm used is the mobilenet-v2 and the sorting threshold for the tracks is 0.3.



(a) 01.09.2023 - 0.25 detection threshold

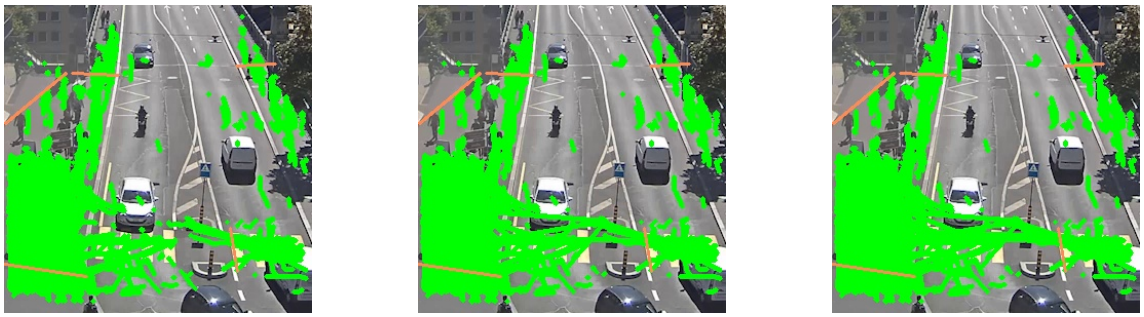


(b) 01.09.2023 - 0.3 detection threshold

(c) 01.09.2023 - 0.5 detection threshold

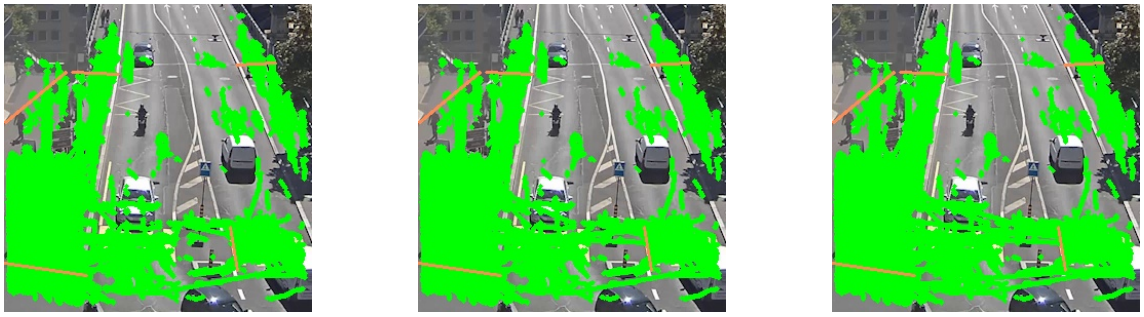
(d) 01.09.2023 - 0.75 detection threshold

A.6 Tracks for video S1



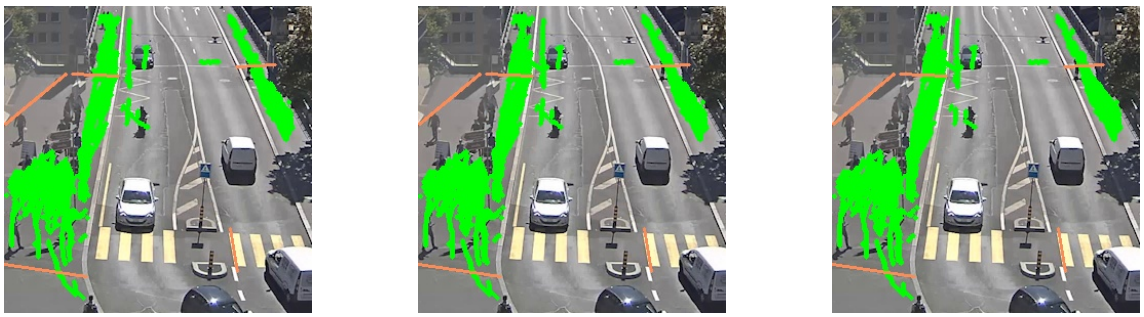
(a) $30f$ (b) $60f$ (c) $90f$

Figure A.4: Visualisation of the tracks of mnet0.5 on video S1



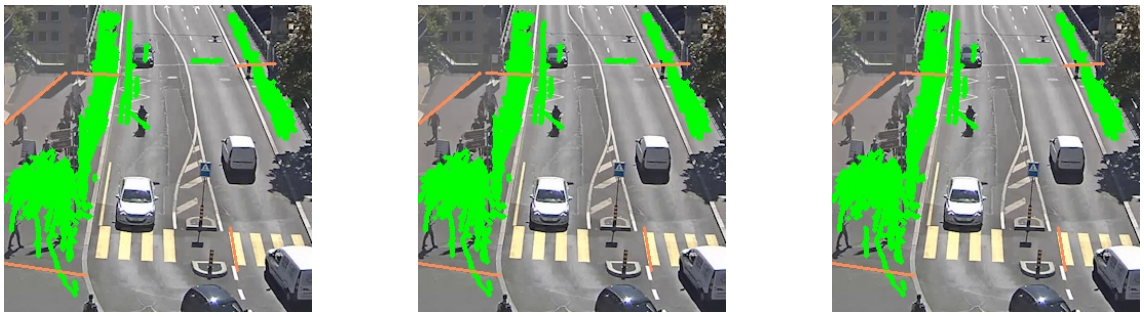
(a) $30f$ (b) $60f$ (c) $90f$

Figure A.5: Visualisation of the tracks of mnet0.35 on video S1



(a) $30f$ (b) $60f$ (c) $90f$

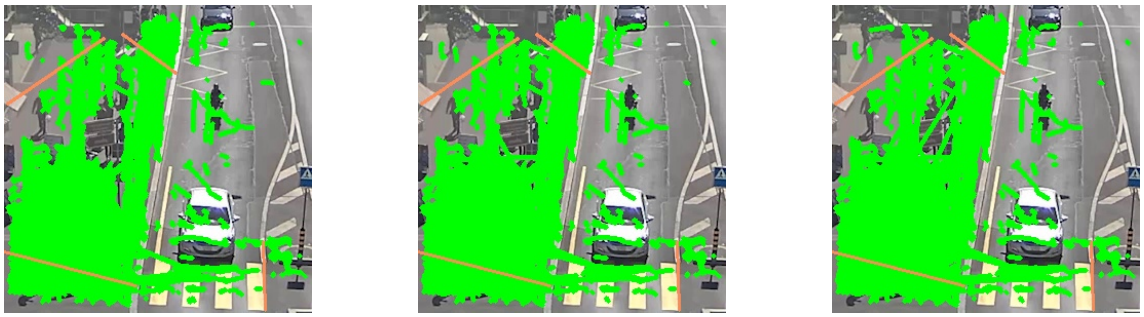
Figure A.6: Visualisation of the tracks of pnet0.5 on video S1



(a) $30f$ (b) $60f$ (c) $90f$

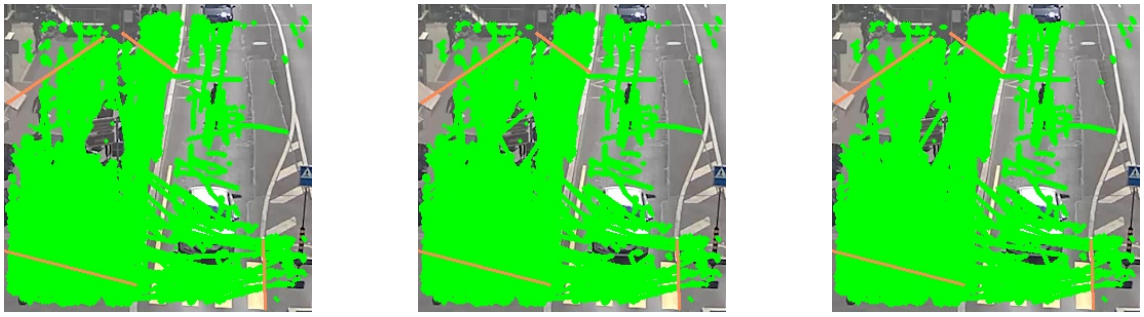
Figure A.7: Visualisation of the tracks of pnet0.35 on video S1

A.7 Tracks for video S2



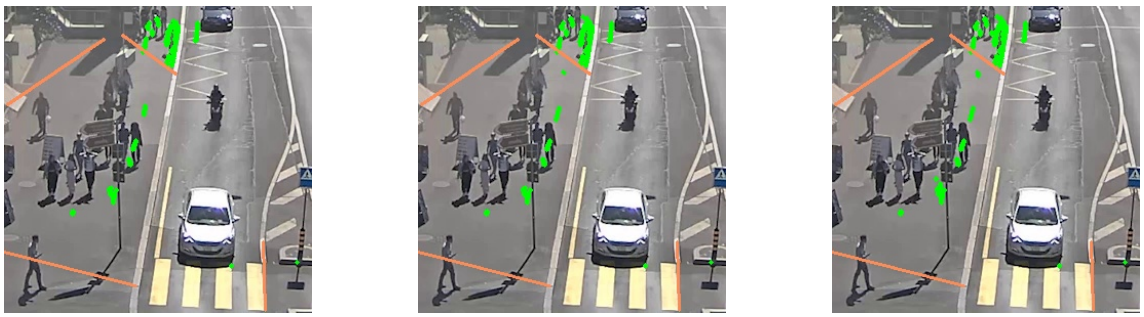
(a) $30f$ (b) $60f$ (c) $90f$

Figure A.8: Visualisation of the tracks of mnet0.5 on video S2



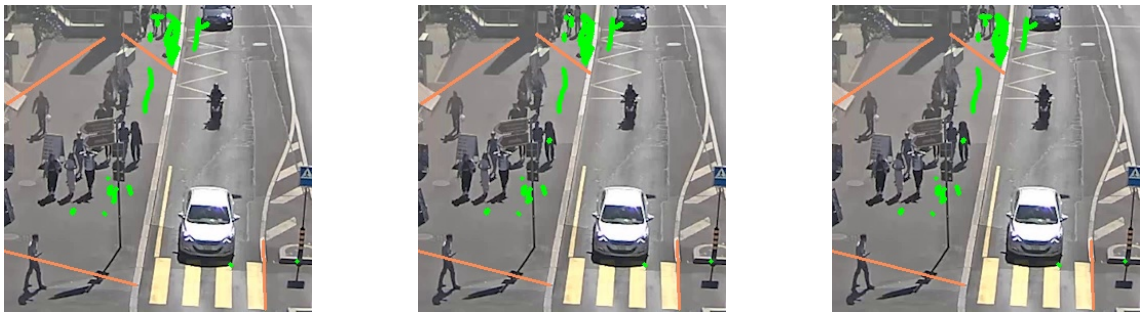
(a) $30f$ (b) $60f$ (c) $90f$

Figure A.9: Visualisation of the tracks of mnet0.35 on video S2



(a) $30f$ (b) $60f$ (c) $90f$

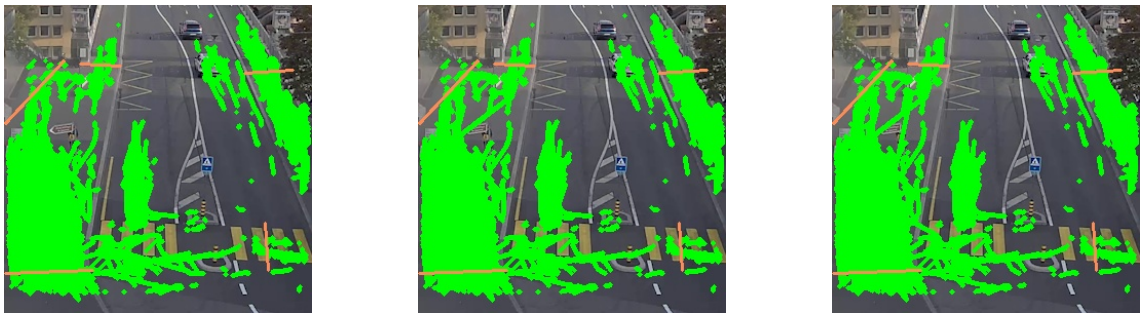
Figure A.10: Visualisation of the tracks of pnet0.5 on video S2



(a) $30f$ (b) $60f$ (c) $90f$

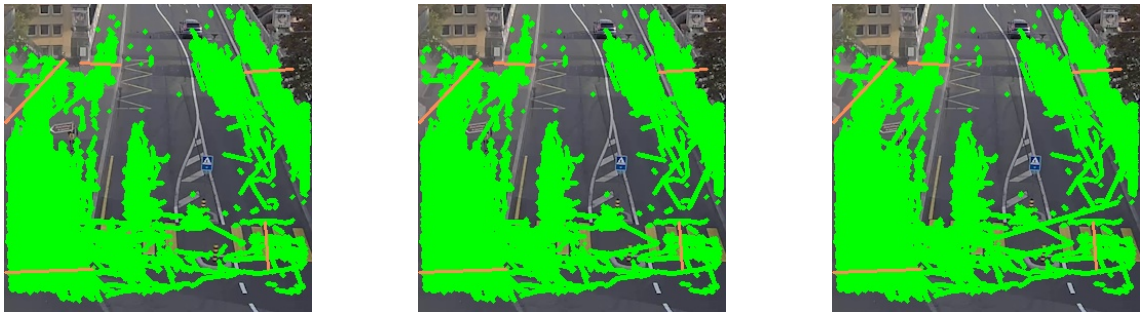
Figure A.11: Visualisation of the tracks of pnet0.35 on video S2

A.8 Tracks for video O1



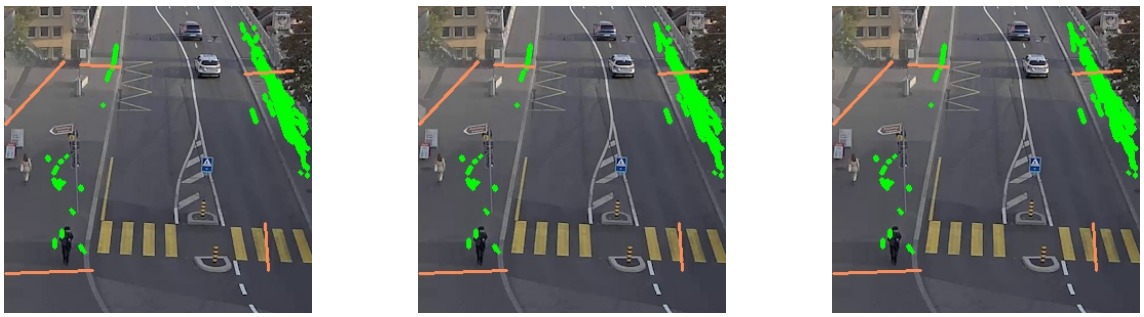
(a) $30f$ (b) $60f$ (c) $90f$

Figure A.12: Visualisation of the tracks of mnet0.5 on video O1



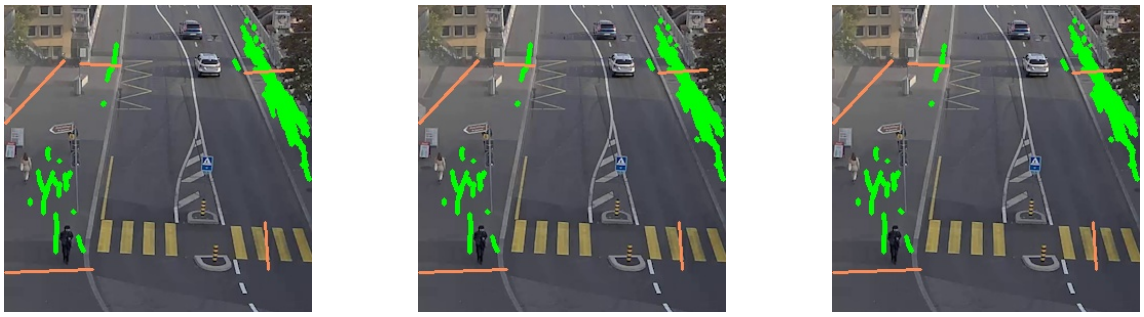
(a) $30f$ (b) $60f$ (c) $90f$

Figure A.13: Visualisation of the tracks of mnet0.35 on video O1



(a) $30f$ (b) $60f$ (c) $90f$

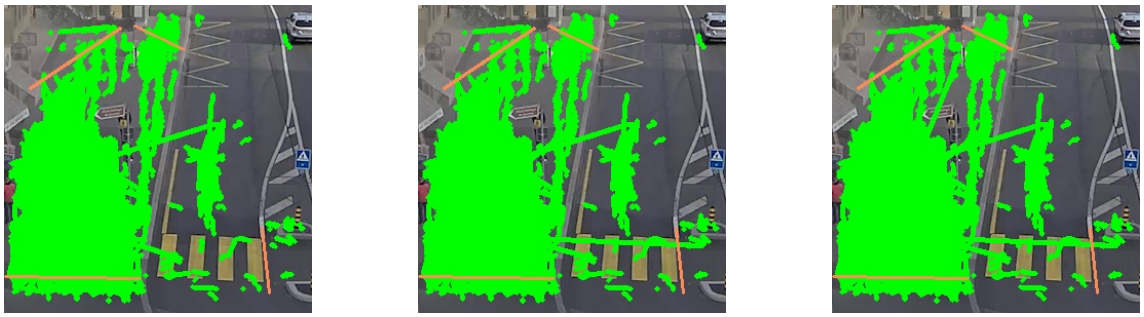
Figure A.14: Visualisation of the tracks of pnet0.5 on video O1



(a) $30f$ (b) $60f$ (c) $90f$

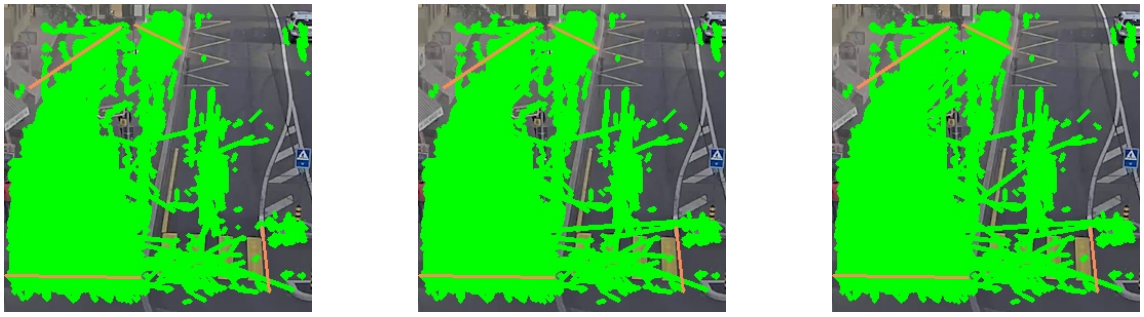
Figure A.15: Visualisation of the tracks of pnet0.35 on video O1

A.9 Tracks for video 04



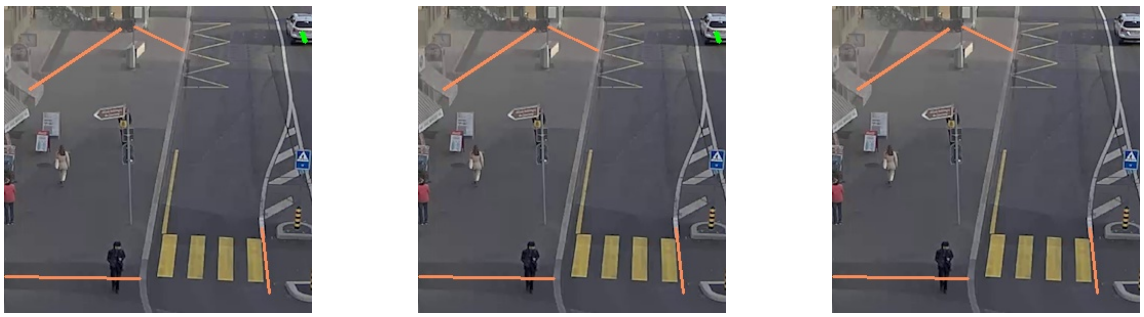
(a) $30f$ (b) $60f$ (c) $90f$

Figure A.16: Visualisation of the tracks of mnet0.5 on video O2



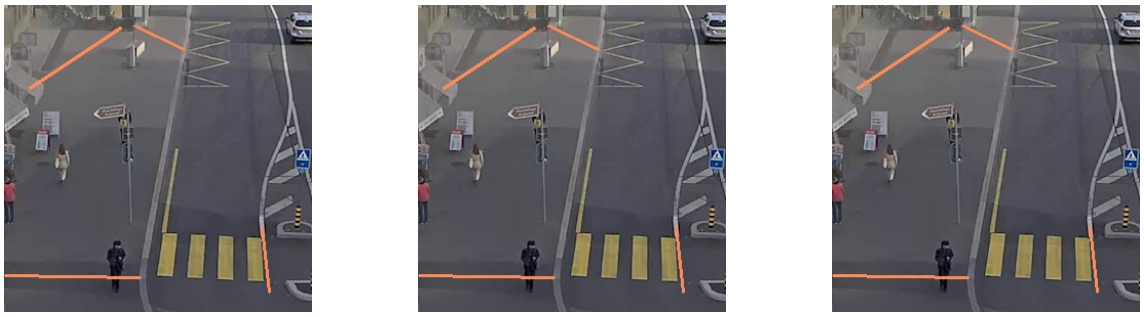
(a) $30f$ (b) $60f$ (c) $90f$

Figure A.17: Visualisation of the tracks of mnet0.35 on video O2



(a) $30f$ (b) $60f$ (c) $90f$

Figure A.18: Visualisation of the tracks of pnet0.5 on video O2



(a) $30f$ (b) $60f$ (c) $90f$

Figure A.19: Visualisation of the tracks of pnet0.35 on video O2

A.10 Origin/Destination tables for video S1

	Origin	Destination
1	0	0
2	8	14
3	12	13
4	1	4
5	111	102

(a) $30f$

	Origin	Destination
1	0	0
2	9	18
3	12	13
4	2	4
5	114	105

(b) $60f$

	Origin	Destination
1	0	0
2	10	19
3	12	13
4	2	5
5	114	109

(c) $90f$

Table A.3: O/D counts for video S1, mnet0.5

	Origin	Destination
1	4	2
2	27	46
3	52	42
4	4	5
5	178	170

(a) $30f$

	Origin	Destination
1	4	2
2	28	50
3	53	42
4	6	7
5	182	174

(b) $60f$

	Origin	Destination
1	4	2
2	29	50
3	53	42
4	8	8
5	179	174

(c) $90f$

Table A.4: O/D counts for video S1, mnet0.35

	Origin	Destination
1	0	0
2	6	21
3	3	1
4	0	0
5	1	0

(a) $30f$

	Origin	Destination
1	0	0
2	7	21
3	4	1
4	0	0
5	1	0

(b) $60f$

	Origin	Destination
1	0	0
2	7	22
3	4	1
4	0	0
5	1	0

(c) $90f$

Table A.5: O/D counts for video S1, pnet0.5

	Origin	Destination
1	0	0
2	7	28
3	6	1
4	0	0
5	1	0

(a) $30f$

	Origin	Destination
1	0	0
2	7	28
3	6	1
4	0	0
5	1	0

(b) $60f$

	Origin	Destination
1	0	0
2	8	26
3	6	1
4	0	0
5	1	0

(c) $90f$

Table A.6: O/D counts for video S1, pnet0.35

A.11 Origin/Destination tables for video S2

	Origin	Destination
1	6	8
2	118	88
3	1	0
4	187	238

(a) $30f$

	Origin	Destination
1	6	9
2	118	90
3	1	0
4	192	246

(b) $60f$

	Origin	Destination
1	7	9
2	118	92
3	1	0
4	195	251

(c) $90f$

Table A.7: O/D counts for video S2, mnet0.5

	Origin	Destination
1	14	14
2	231	194
3	3	6
4	325	415

(a) $30f$

	Origin	Destination
1	15	15
2	244	204
3	3	5
4	328	425

(b) $60f$

	Origin	Destination
1	15	16
2	247	208
3	3	5
4	331	428

(c) $90f$

Table A.8: O/D counts for video S2, mnet0.35

	Origin	Destination
1	0	0
2	1	0
3	0	0
4	0	0

(a) $30f$

	Origin	Destination
1	0	0
2	1	0
3	0	0
4	0	0

(b) $60f$

	Origin	Destination
1	0	0
2	1	0
3	0	0
4	0	0

(c) $90f$

Table A.9: O/D counts for video S2, pnet0.5

	Origin	Destination
1	0	0
2	0	0
3	0	0
4	0	0

(a) $30f$

	Origin	Destination
1	0	0
2	0	0
3	0	0
4	0	0

(b) $60f$

	Origin	Destination
1	0	0
2	0	0
3	0	0
4	0	0

(c) $90f$

Table A.10: O/D counts for video S2, pnet0.35

A.12 Origin/Destination tables for video O1

	Origin	Destination
1	6	3
2	6	6
3	79	53
4	0	1
5	68	82

(a) $30f$

	Origin	Destination
1	7	3
2	6	6
3	79	54
4	0	1
5	67	84

(b) $60f$

	Origin	Destination
1	7	3
2	6	6
3	80	54
4	0	1
5	67	86

(c) $90f$

Table A.11: O/D counts for video O1, mnet0.5

	Origin	Destination
1	13	10
2	11	15
3	135	100
4	1	2
5	92	117

(a) $30f$

	Origin	Destination
1	13	10
2	12	15
3	135	101
4	1	3
5	93	120

(b) $60f$

	Origin	Destination
1	13	12
2	12	15
3	136	101
4	1	3
5	93	120

(c) $90f$

Table A.12: O/D counts for video O1, mnet0.35

	Origin	Destination
1	0	0
2	1	0
3	4	1
4	0	0
5	0	0

(a) $30f$

	Origin	Destination
1	0	0
2	1	0
3	5	1
4	0	0
5	0	0

(b) $60f$

	Origin	Destination
1	0	0
2	1	0
3	4	1
4	0	0
5	0	0

(c) $90f$

Table A.13: O/D counts for video O1, pnet0.5

	Origin	Destination
1	0	0
2	1	0
3	4	3
4	0	0
5	0	0

(a) $30f$

	Origin	Destination
1	0	0
2	1	0
3	4	2
4	0	0
5	0	0

(b) $60f$

	Origin	Destination
1	0	0
2	1	0
3	5	2
4	0	0
5	0	0

(c) $90f$

Table A.14: O/D counts for video O1, pnet0.35

A.13 Origin/Destination tables for video O2

	Origin	Destination
1	4	4
2	17	30
3	0	0
4	76	117

(a) $30f$

	Origin	Destination
1	5	4
2	18	31
3	0	1
4	76	118

(b) $60f$

	Origin	Destination
1	6	4
2	18	31
3	0	1
4	76	119

(c) $90f$

Table A.15: O/D counts for video O2, mnet0.5

	Origin	Destination
1	16	11
2	60	74
3	0	2
4	111	157

(a) $30f$

	Origin	Destination
1	16	11
2	63	80
3	0	2
4	116	159

(b) $60f$

	Origin	Destination
1	16	12
2	63	83
3	0	2
4	118	160

(c) $90f$

Table A.16: O/D counts for video O2, mnet0.35

	Origin	Destination
1	0	0
2	0	0
3	0	0
4	0	0

(a) $30f$

	Origin	Destination
1	0	0
2	0	0
3	0	0
4	0	0

(b) $60f$

	Origin	Destination
1	0	0
2	0	0
3	0	0
4	0	0

(c) $90f$

Table A.17: O/D counts for video O2, pnet0.5

	Origin	Destination
1	0	0
2	0	0
3	0	0
4	0	0

(a) $30f$

	Origin	Destination
1	0	0
2	0	0
3	0	0
4	0	0

(b) $60f$

	Origin	Destination
1	0	0
2	0	0
3	0	0
4	0	0

(c) $90f$

Table A.18: O/D counts for video O2, pnet0.35